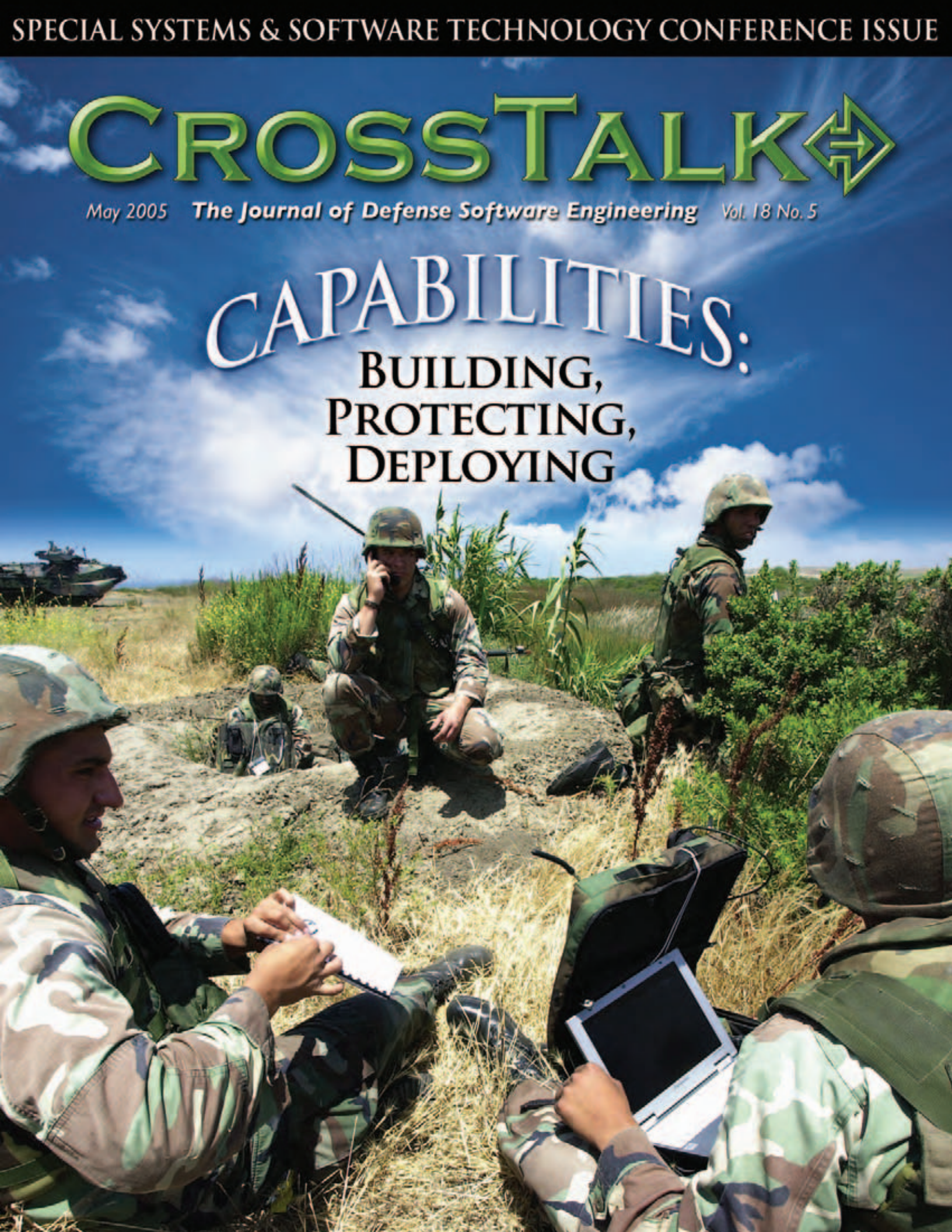
SPECIAL SYSTEMS & SOFTWARE TECHNOLOGY CONFERENCE ISSUE

CrossTalk

May 2005 The Journal of Defense Software Engineering Vol. 18 No. 5

CAPABILITIES: BUILDING, PROTECTING, DEPLOYING



Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
MAY 2005		**00-00-2005 to 00-00-2005**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
CrossTalk: The Journal of Defense Software Engineering. Volume 18, Number 5, May 2005	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **OO-ALC/MASE,6022 Fir Ave,Hill AFB,UT,84056-5820**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

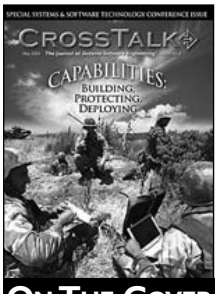
14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**32**	

Policies, News, and Updates

Software Engineering Technology



On The Cover
Cover Design by Kent Bingham.

Departments

CrossTalk

OC-ALC/ MAS Co-Sponsor	*Kevin Stamey*
OO-ALC/MAS Co-Sponsor	*Randy Hill*
WR-ALC/MAS Co-Sponsor	*Tom Christian*
Publisher	*Tracy Stauder*
Associate Publisher	*Elizabeth Starrett*
Managing Editor	*Pamela Palmer*
Associate Editor	*Chelene Fortier-Lozancich*
Article Coordinator	*Nicole Kentta*
Creative Services Coordinator	*Janna Kay Jensen*
Phone	(801) 775-5555
Fax	(801) 777-8069
E-mail	crosstalk.staff@hill.af.mil
CrossTalk Online	www.stsc.hill.af.mil/crosstalk

Oklahoma City-Air Logistics Center (OC-ALC), Ogden-Air Logistics Center (OO-ALC), and Warner Robins-Air Logistics Center (WR-ALC) MAS Software Divisions are the official co-sponsors of CrossTalk, The Journal of Defense Software Engineering. The MAS Software Divisions and the Software Technology Support Center (STSC) are working jointly to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

The STSC is the publisher of CrossTalk, providing both editorial oversight and technical review of the journal.

S T S C

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail us or use the form on p. 30.

OO ALC/MASE
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at <www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf>. CrossTalk does not pay for submissions. Articles published in CrossTalk remain the property of the authors and may be submitted to other publications.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with CrossTalk.

Trademarks and Endorsements: This Department of Defense (DoD) journal is an authorized publication for members of the DoD . Contents of CrossTalk are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, or the STSC. All product names referenced in this issue are trademarks of their companies.

Coming Events: Please submit conferences, seminars, symposiums, etc. that are of interest to our readers at least 90 days before registration. Mail or e-mail announcements to us.

CrossTalk Online Services: See <www.stsc.hill.af.mil/crosstalk>, call (801) 777-7026, or e-mail <stsc.webmaster@hill.af.mil>.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

Dr. Mom's Approach to Improved Capabilities

"*Over* budget, *behind* schedule." These are words that send chills through a customer's spine and words that no one acquiring systems wants to hear; however, they are words that are deeply ingrained and heard all too frequently in the culture of systems development.

If you're like me, you have been in countless meetings where project managers have reported their programs to be either over-expensing or past the target date. Such announcements also seem to be followed by a get-well plan. With nearly 40 percent of all embedded software development projects reported to be behind schedule, it appears there are an enormous number of these infamous get-well plans.

This year's Systems and Software Technology Conference (SSTC 2005) runs from April 18-21. The focus of SSTC 2005 is on showcasing new tools to help improve the capabilities we provide our customers. Throughout the conference, attendees concentrate on acquiring new systems as well as building and delivering better systems, and highlighting future concepts such as automated code restructuring. They also explore how modern networking is being used on the battlefront. The ideas presented at SSTC 2005 center on the theme of building and delivering needed capabilities to customers, ultimately the warfighters.

I'd like to compare SSTC this year to a steaming bowl of mom's homemade chicken soup: designed to soothe the ills of any sick program. As you read through this month's articles, which are compiled from a few of the excellent papers presented at SSTC, you will note that in particular Dr. (not the medical kind) Richard Turner's article about the Department of Defense's (DoD) Acquisition Best Practices Clearinghouse uses a medical analogy to illustrate some acquisition best practices. With so many sick programs out there, let's continue this medical metaphor to describe what else is in this month's CrossTalk.

You will find great prescriptions for implementing DoD regulation, policies, and mandates such as Technology Readiness Assessments mandated by the DoD 5000 regulations series, and an article by David R. Basel, Dana Foat, and Cragin Shelton describing requirements regarding the Ports, Protocols, and Services Management Process. For many of us, implementing these types of requirements is like having mom suddenly appear above you with a teaspoon, filled to capacity with the most vile, dreaded fluid imaginable – cough medicine, which she demands we take, all because of being unable to stifle that single pillow-muffled cough that her super-human hearing detected through a closed door. Just thinking about that nasty stuff makes my taste buds cringe – YUCK!

Also in this month's lineup are articles by Robert A. Martin, Dr. David A. Cook, Dr. James M. Skinner, and Paul E. McMahon that address some formidable issues, which when implementing these great ideas, at times feels like swallowing large, bitter pills. Drs. (also, not the medical kind) Cook and Skinner delve into good uses of modeling and simulation in creating software. They explore how models have grown in complexity and give some techniques of verification and validation that will help improve modeling life-cycle costs. McMahon helps diagnose what agile software development is and how to extend agile development methods to large dispersed environments. He also dispels some old wives' tales and myths regarding agile. Martin's article helps triage issues of managing transformational vulnerabilities with standards.

Lastly, Robert Gold and David Jakubek discuss the preventative health measures of Technology Readiness Assessments (TRA) when conducted on major defense acquisitions. Prior to system development, a TRA documents that there is a reasonable expectation that the acquisition is technically feasible.

Now, I hope my descriptions haven't scared you away from reading these great articles. I promise reading them won't be like going to the doctor (yes, the medical kind). Every article has some great ideas for improving the overall health of programs and projects. I do hope you will enjoy this dose of CrossTalk. It may not be sugar-coated, but just like Mom would say about a little chicken soup, "It's good for you!"

Brent D. Baxter
SSTC Technical Conference Manager
Manager, Software Technology Support Center

Introducing the Department of Defense Acquisition Best Practices Clearinghouse

Kathleen Dangle
Fraunhofer Center, University of Maryland

Laura Dwinnell
Northrop Grumman Information Technology

John Hickok
Defense Acquisition University

Dr. Richard Turner
Systems and Software Consortium

Thursday, 21 April 2005
Track 3: 11:30 – 12:15 p.m.
Ballroom C

This new capability will help acquisition personnel identify, select, and implement validated practices according to their programmatic needs.

Acquisition best practices are much like medicines. They are indicated for certain problems and provide specific benefits. They can sometimes be harmful if used improperly, in combination with other practices, or in the wrong situation. Practices take differing amounts of time to produce their benefits. There can be various levels of certification or caveats. There are usually specific instructions (like dosages or usage information). There is definitely a cost to be weighed against the benefit. And, finally, even with best practices, you sometimes need different levels of expert advice to help select and implement them.

What if choosing a practice were more like choosing a medicine? You could make your choice based on your needs and reliable information. To do this, practices would need to be described like medicines – in a uniform, brief way so that you can compare information and make educated choices without reading a ton of material. Medical information is backed up by a lot of studies, clinical trials, approvals, and usually, good science – but you do not need to check out all that research. Of course, if you want some additional information, your pharmacist, doctor, or favorite medical Web site probably can provide all the nitty-gritty details you desire.

This is exactly the approach taken by the Department of Defense (DoD) Acquisition Best Practices Clearinghouse (BPCh). It will provide brief, useful descriptions of practices and their characteristics based on carefully analyzed data. Users can access the information according to their preferences and needs. The BPCh will also provide the trail of evidence used to establish the characteristics as well as advice on where to find out more detailed information. With this information, users can answer a wide range of questions such as the following:

- Where can I find measurement practices that will work for my program?
- How do I find out if the accelerated life testing practice my friend recommended is useful or just hype?
- I want to add inspections to improve my software quality. Are they worth the cost and if so, what is a good first step?

"The BPCh comprises an integrated set of processes, tools, data, and people that maintain a continuously improving resource of best practices information ..."

- I have taken over an acquisition program just after milestone B. What practices should I look for in my contractors?
- Our organization is implementing the Capability Maturity Model® Integration (CMMI®) Acquisition Module. What practices should I implement to satisfy the project monitoring and control goals?
- What practices can help me meet the Office of the Secretary of Defense's (OSD) new policy on system engineering plans?

Sponsored by the Defense Acquisition University (DAU), the assistant secretary of defense for Network and Information Integration, and the under secretary of defense for Acquisition, Technology and Logistics (AT&L), the BPCh will be a single source for identifying, selecting, and implementing validated acquisition-related practices based on specific program needs. The Fraunhofer Center at the University of Maryland is developing the concept and processes, Northrop Grumman Information Technology will build the operational system, and DAU will host and continue to evolve the capability as an integrated part of OSD's knowledge-sharing systems.

The BPCh comprises an integrated set of processes, tools, data, and people that maintain a continuously improving resource of best practice information that is readily accessible via a wide range of methods. Figure 1 shows its operational architecture, roles, and four basic processes: Best Practice Contributions, Best Practice Handling, BPCh Usage, and BPCh Operations.

The *Best Practice Contributions* process provides a way for users to nominate practices for inclusion in the BPCh, or to provide additional data on current BPCh practices.

The *Best Practice Handling* process is what makes the BPCh special. It distills information about practices – lessons learned, research reports, measurement data – into a practice profile that is easy to understand and work with. Of course, connectivity to the source material is preserved. Once created, these profiles are maintained in a repository and serve as the basis for recommending and selecting practices. This process includes five primary phases:

® Capability Maturity Model and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

- **Identification.** Best practice suggestions are submitted by information providers as studies, lessons learned, or other data. Leads will be collected, reviewed, categorized, consolidated, and prioritized into a continuously updated ranked list of candidate practices.
- **Quantification and Qualification.** The highest priority candidate practice is investigated to gather more data, build stories, and generate more detailed practices if appropriate. This results in better defined candidate practices ready for characterization. Practices may be tabled until more information is available.
- **Analysis and Synthesis.** This step defines the practice profile attributes, builds a business case, and captures any interrelationships with existing practices. The result is a practice package.
- **Validation.** The set of practice packages is reviewed by a panel of experts in the relevant topic. Approved practices move on to the next phase. The panel may, however, send packages back to previous phases.
- **Packaging and Dissemination.** The validated best practices are published within the BPCh repository, simplified implementation guidance may be developed, and information about the practice may be communicated to users via a wide range of media.

A recommended practice must successfully pass through the Best Practice Handling process. Initially, there will be a number of valuable practices with insufficient data available to pass. There will also be new practices identified that may be promising enough for the BPCh to maintain information about, but not necessarily recommend to average programs. For these reasons, practices will be characterized at a number of recommendation levels. It is also possible that practices may be identified as *to be avoided* under specific conditions.

The Fraunhofer Center is developing processes and tools to support Best Practice Handling. The National Defense Industrial Association, International Council on Systems Engineering, and Software Engineering Institute, along with academic and defense organizations, will support both Best Practice Contributions and Best Practice Handling processes.

BPCh Usage is the process that helps users identify and access information about practices. Much thought is going into the usefulness and usability of the BPCh, particularly user access to its content. There will be numerous ways to locate practices, including searching and browsing. A number of search methods, simple and complex, will be provided. Multi-viewpoint browsing will let the user look at practices organized in the way that is most useful to them. Examples of viewpoints include the following:

- DoD 5000 program phases and milestones.
- System Engineering Plan sections.
- CMMI structure (staged and continuous).
- Discipline (e.g., systems engineering, software acquisition).
- AT&L functions/topics (e.g., program manager, source selection).
- Program risks.

Due to the diverse nature of user styles and information needs, the BPCh will allow users to maintain user and project profiles that can be used to filter the practices presented and tailor agents that notify the user of changes in the repository or other BPCh events.

The *BPCH Operations* process ensures the infrastructure required for the BPCh to operate is maintained. With DAU operating the BPCh, it will be closely coupled to the DoD acquisition knowledge infrastructure with access both to and from the AT&L Knowledge Sharing System, "Defense Acquisition Guidebook," Acquisition Community Connection, and other knowledge communities, as well as DAU's Learning Asset Repository. Information on these can be found at <http://akss.dau.mil> and <http://acc.dau.mil>.

An initial BPCh prototype is being demonstrated at the April 2005 System and Software Technology Conference in Salt Lake City, Utah. An advanced prototype for limited operational evaluation is scheduled for demonstration at the NDIA Systems Engineering Conference in October 2005. Full online system operation and public access is planned for fall of 2006.

The development team is working closely with user groups to capture scenarios and identify the most useful information content, functionality, and access and display methods. The more potential users who provide input, the better the interface will be. If you are interested in participating in a user group, as a user, or as a practice provider, please contact the project at <BPCH@fc-md.umd.edu> or <http://acc.dau.mil/bpch>.◆

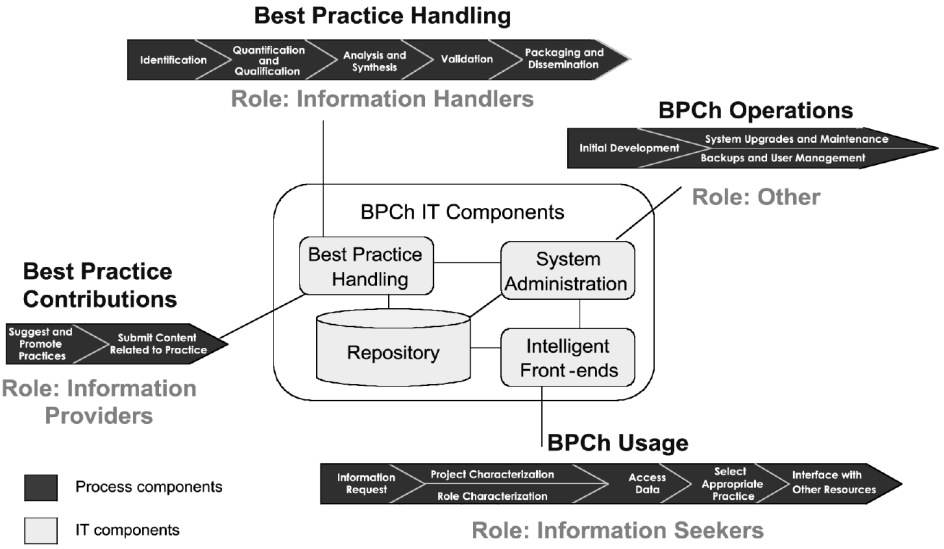


Figure 1: *Best Practices Clearinghouse Components*

Visit Us at SSTC

Stop by the BPCh booth No. 118 at SSTC 2005 and try out the prototype. Your feedback will help make the BPCh a more useful and usable tool for your needs.

About the Authors

Kathleen Dangle
Fraunhofer Center
University of Maryland
E-mail: kdangle@fc-md.umd.edu

Laura Dwinnell
Northrop Grumman
Information Technology
E-mail: laura.dwinnell@ngc.com

John Hickok
Defense Acquisition University
E-mail: john.hickok@dau.mil

Richard Turner, Ph.D.
Systems and Software Consortium
E-mail: rich.turner.ctr@osd.mil

Ports, Protocols, and Services Management Process for the Department of Defense

David R. Basel
Defense Information Systems Agency

Dana Foat
Defense-Wide Information Assurance Program Office

Cragin Shelton
The MITRE Corporation

Tuesday, 19 April 2005
Track 1: 4:50 – 5:35 p.m.
Ballroom A

All automated information systems (AIS) used on Department of Defense (DoD) data networks must register the data communication modes identifying the ports, protocols, and application services (PPS) used, and the network boundaries crossed. Compliance with the PPS requirements will reduce development time and cost, increase security, speed certification and accreditation steps, enhance AIS interoperability across the department, and speed operational deployment of all new and updated AIS in DoD. This article introduces the PPS basic concepts, and demonstrates how developers and program managers can comply with the PPS requirements, leverage the security analysis provided by the management office, and obtain the benefits listed.

The overall goal of the Ports, Protocols, and Services Management Process (PPSMP) [1] is to improve both the interoperability of joint applications and the security of the overall Department of Defense (DoD) information infrastructure. The process supports many people in many roles: program managers, systems engineers, software developers, network operators, network security managers, router and firewall administrators, etc.

The authors hope this article reaches many of those people, and helps them understand both the value of participating in the PPSMP and the services it can provide.

For reader clarification, we begin this article with a discussion of the basic terms and concepts of computer network traffic on the Internet: protocol, Internet protocol (IP), IP protocol, service/application protocol/data service, and port.

Protocol

A protocol is simply an agreed upon way to communicate or interact. This word can have multiple meanings in different contexts. The meaning and context will become apparent below.

Internet Protocol

The most fundamental protocol is the IP. The international group Internet Engineering Task Force (IETF) [2] sets the standards for the IP. The IETF's many standards documents – both draft and approved – called "Requests for Comment (RFC)," are at <www.ietf. org>.

The core principal of the IP is that all information travels between computers in data bundles called packets rather than in a continuous stream. Any information, for example a computer file, can be divided into packets by the sending computer and the packets reassembled into the complete file by the receiving computer. The IP defines several structures so this can happen. The two most important are an addressing scheme and a defined packet structure. Figure 1 illustrates the IP communications concept.

An IP address is like a computer's telephone number. You may have seen these four groupings of numbers separated by periods (or dots). Those are IP addresses as defined under Version 4 of the IP standard currently in use internationally.

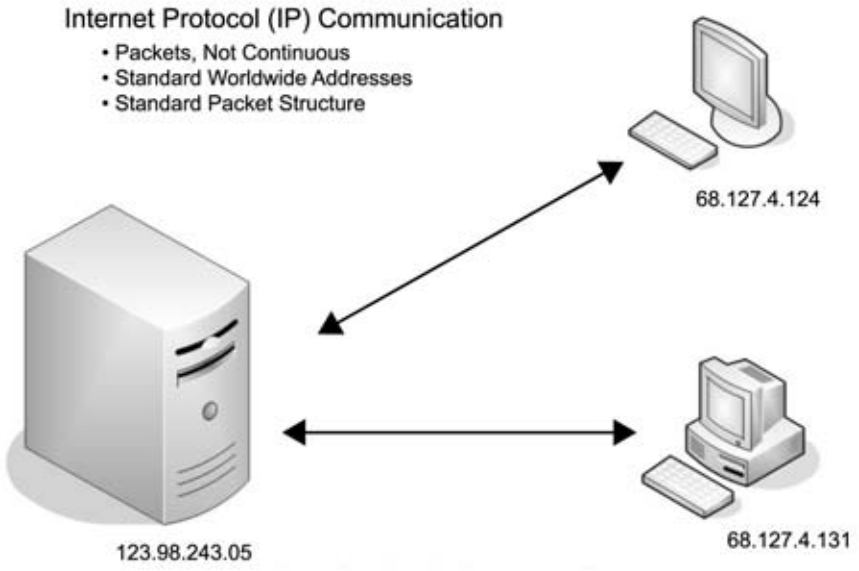
Each of the four number groups can be any number from 0 to 255. Many countries, as well as the DoD, are upgrading to IP Version 6 (IPv6), which uses a very different address scheme. To save space, this article does not discuss IPv6 addressing. Many resources on IPv6 are available for the readers[1].

Once two computers find each other, they need to agree on how to communicate. The defined packet structure assists in this process. Very simply, each packet has two major parts: the header, with information about the packet, and the payload, with the actual data. Figure 2 illustrates this concept.

More than 100 standard rule sets, each with a different purpose, are available for the computer-to-computer communication. Here are examples of a few of the IP protocols most often seen on networks:

- **Protocol 1: ICMP - Internet Control Message Protocol.** ICMP packets allow two operating systems to trade status messages. A ping is a single packet sent from one computer to another that means simply, "Are you there?" If the receiving computer is listening for the ping and chooses to admit it is available for further contact,

Figure 1: *IP Communication*



it replies with a one-packet ping reply, which tells the first computer, "Yes, I'm here and listening."

Why is it named *ping*? The name is borrowed from the world of submarines and sonar. Think back to every submarine movie you have seen. The sound sent out by the sonar is called a ping, because of the way it sounds on the speakers when reflected back to the submarine by the enemy ship lurking in the distance.

- **Protocol 6: TCP - Transmission Control Protocol.** TCP packets support positive confirmation that each chunk of data (packet) arrived intact and unchanged. This confirmation is essential when sending data that must not be lost or corrupted, such as a database record.
- **Protocol 17: UDP - User Datagram Protocol.** UDP packets can carry any content data, but they do not provide a feedback mechanism to confirm receipt of intact data. UDP is used to push large amounts of data (packets) out from the computer, and it is not critical if some of them get lost or broken, such as streaming audio or streaming video.
- **Protocol 50: ESP - Encapsulating Security Payload.** ESP packets have the primary data encrypted; only the sender and receiver, who have the right encryption keys, can read the data. This protection of confidentiality is part of the IP security set of standards, and is the basis for many virtual private networks.

Here is jargon watch No. 1: *Protocol* is the first word in this discussion with multiple meanings. The IP defines the overall structure of packets and IP addresses. The IP protocol defines the major type and function of packets. (Yes, spelling out *IP protocol* as *Internet protocol protocol* does sound redundant, but it is accurate. That is why you never see it spelled out.) The next term to be discussed, *service*, is also called the application protocol. In the context of the PPSM, protocol most often refers to the IP protocol.

Service

Once two computers are communicating with the proper hardware system addresses (IP address), and computer-to-computer packet type (IP protocol), the individual programs on each computer still need to communicate properly. The programs must exchange data in the right format and packet structure for the programs themselves to understand. The rule for the agreed format at this level is called variously the *application protocol*, *data service*, or simply *service*. All three terms refer to the same aspect of the packet.

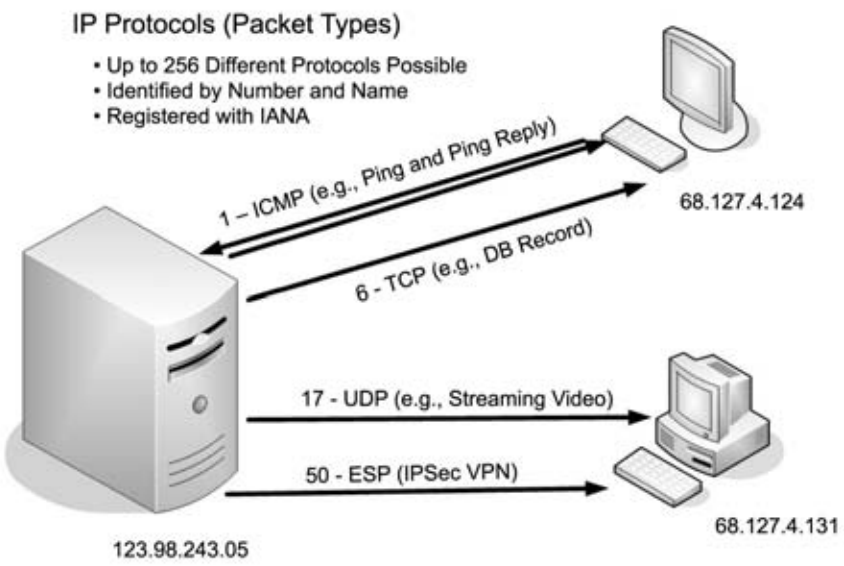


Figure 2: IP *Protocols*

While the IP has provisions for up to only 256 different protocols, it has set no limit on the number of possible services. Every application programmer could devise unique services for the programs to use to communicate over a network. In the interests of both interoperability and ease of programming, most do not. However, new programs introduce new services every year. Some are proprietary, while others become widely used standards.

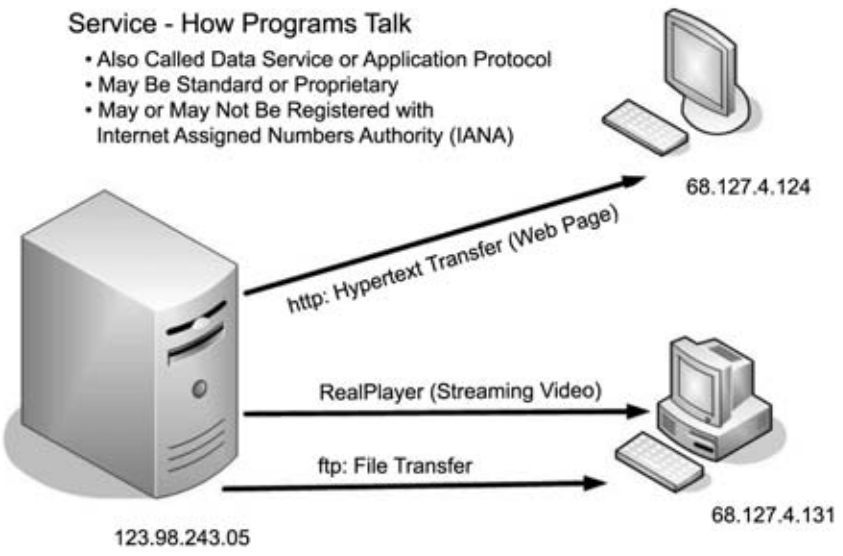
There are thousands of these services or application protocols because different types of programs need different types and orders and formatting for their data. Programmers who hope to see an application protocol become a standard may publish the specification as a RFC with the IETF, but they are not required to do so. Figure 3 illustrates the use of these services.

Examples of well-known and commonly used services include hypertext transfer protocol (http), used for sending Web pages; file transfer protocol (ftp), used for moving entire data files between computers; and telnet, used for remote computer terminal connections.

Within the packet-structure logic described so far, generally only two protocols carry packets that contain a defined service at the next level: TCP and UDP. In fact, many services can travel in either. Program architects decide which protocol to use depending on whether the priority is higher speed or packet-count throughput (UDP), or 100 percent intact packet arrival (TCP).

Here is jargon watch No. 2: Within the PPSM, *service* refers to the data service or

Figure 3: *Service*



application protocol as just described. However, in the world of network engineering, *service* can refer to a level or quality of service of the network (actual data transport speed limit and network availability). Service can also refer to a category of application running on a computer such as a Web service, a file service, a time reference service, a chat service, and so forth. A computer running one or more services is called a *server* (Web server, file server, time server, or chat server, for example).

Port

The IP address sends each packet to the right computer. The IP protocol tells the receiving computer the packet type. The service indicated in the packet tells the receiving program the data structure. However, each computer may have many different programs running at the same time. And each program may be having conversations with more than one other computer at the same time.

For instance, a file server may be receiving thousands of requests to send out files to individual remote computers in the same minute. If the server agrees to support all of those requests, it needs some way to sort the incoming traffic to keep track of each two-way conversation, or session, separately.

The concept of the port provides the tool for managing multiple simultaneous sessions. For TCP and UDP packets with defined services, each packet also specifies a port number. The standard packet structure reserves enough room for the port number so it could be any number from 0 to 65,535.

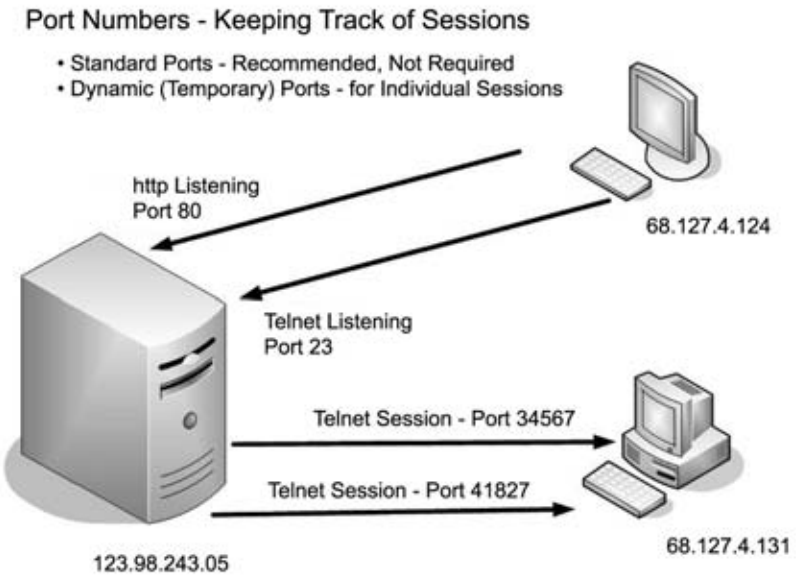
Think of each computer as an office building with one street address (the IP address), but many mailboxes for the separate offices. Think of the port number as the number on each of those internal mailboxes. Some offices (programs) by agreement use a standard local mailbox (port) number. For example, bulk mail delivery goes to port 25, requests for Web pages go to port 80, and requests for terminal sessions (telnet) go to port 23. Figure 4 illustrates the use of ports.

To help keep multiple conversations separate once a contact is started the host may say, "Send everything else for this session to one of my high number boxes. This is temporary, just for this exchange." These are dynamic or transient (or ethereal) ports.

By convention and as defined by both the IETF and the Internet Assigned Numbers Authority (IANA) [3], port numbers from 0 to 1,023 should be used only for the standard services as registered with IANA. Other services should use port numbers above 1,024. Developers may, if they wish, register with IANA any proprietary or non-standard service's use of particular ports above 1,024. This does not really reserve those ports for only that use, but it does notify all network users and engineers of the planned use.

Here is jargon watch No. 3: In the context of the PPSM and IP packet headers, *port* refers to the number that helps manage communication sessions. This information is important to network engineers and administrators who manage routers and firewalls, as described in the next section. However, those same router and firewall administrators also use port to refer to the physical connection on hardware where they plug in a network cable.

Figure 4: *Ports*



Boundary Filtering - Routers and Firewalls

So, why does the PPSM care about all of those standards? Because network administrators can use them to enforce rules on network traffic via routers and firewalls. Figure 5 illustrates the use of firewalls. These rules can help limit traffic, block problem traffic, and allow favored traffic. Many network management and security devices can use the information in packet headers to decide how to handle the packets. Remember, each packet begins with a header, which is like an envelope, containing the following information: IP address from, IP address to, IP protocol (packet type), service type (if needed), port number (if needed).

Administrators can devise and apply rules based on header items, as follows:

- Deny any traffic that comes from a particular IP address or range of addresses.
- Deny any traffic that uses a service for a program that only insiders should be using.
- Allow any traffic using the standard port for a standard service (application protocol).
- Deny any traffic using a nonstandard port for a service declared.

Policy and Process

The purpose of the ports and protocols policy is to provide the DoD with a framework for managing the use of PPS implemented within DoD information systems. This allows network administrators to know what data types are expected on their networks. At the same time it provides information on what types of traffic to block to protect the network.

The PPS policy, DoD Instruction 8551.1 [1], states that PPS that are visible to DoD-managed network components shall undergo a vulnerability assessment, be assigned to an assurance category, be appropriately registered, be regulated based on its potential to cause damage to DoD operations and interests if used maliciously, and be limited to only the PPS required to conduct official business. In this section, we will discuss the processes of vulnerability assessments and assurance category assignments, the regulation of use or elimination of PPS within the DoD space, and the registration of information systems.

Vulnerability Assessments

Vulnerability assessments identify the security limitations of PPS. Any known countermeasures to limit the exposure of the vulnerability are identified in this

process. This research is performed by the Technical Advisory Group (TAG), which consists of subject matter experts from all of the DoD components, supported by technical expertise from the companies EDS and NetSec under the Defense Information Systems Agency (DISA) Information Assurance [4] contract.

Because of the complexity of some proprietary protocols, it may be necessary to invite service providers to explain the operation of their protocols. The vulnerability assessment reports are available at <http://iase.disa.mil/ports/index.html>.

Assurance Category Assignments

Assurance category assignments identify the relative strength of PPS. The guidance discourages the use of low assurance PPS lacking adequate security countermeasures (category Red); accepts the use of medium assurance PPS, provided documented countermeasures are implemented (category Yellow); and encourages using high assurance PPS, which is considered a best practice when documented countermeasures are implemented (category Green).

Although this research is performed by the TAG, it is verified by the Configuration Control Board (CCB) to ensure that proposed countermeasures may be implemented in an operational network environment. The results of this process are documented in the PPS assurance Category Assignments List (CAL) [5].

Regulation

The regulation of the PPS in DoD network space is a Defense Information System Network Security Accreditation Working Group (DSAWG) decision based on the recommendation of the CCB and the vulnerability assessment reports. The goal is to allow only those PPS that are required to conduct official business to cross enclave boundaries. In general, any PPS labeled as Red must not cross any enclave boundary into the DoD network space.

After issuance of the DSAWG decision, users of information systems implemented with Red PPS will be notified and the DoD PPS program manager (PM) will assist them with compliance. Within the two-year compliance timeframe, the information system may be redesigned with a more secure PPS (Yellow or Green), modified to alter the communications path, or implemented within a Virtual Private Network (VPN) solution.

Although a PPS may be labeled as Red, that category assignment does not necessarily mean that its use will be immediately eliminated from the DoD network space. With the assistance of the DoD PPS program manager, a DoD component may request an appeal for the implementation of the PPS within the information system to the DoD chief information officer. For some PPS designated as Red, there may be no more secure alternatives.

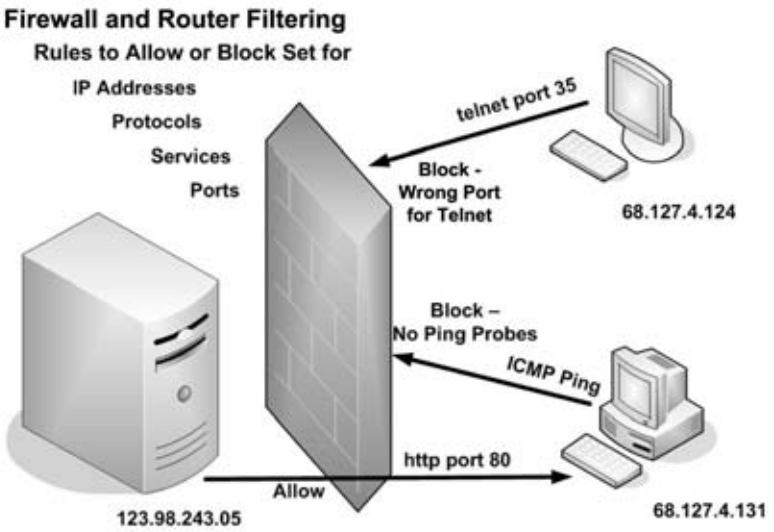


Figure 5: *Firewall and Router Filtering*

The TAG will review these PPS periodically to determine if new countermeasures provide adequate protections or if more secure alternative solutions become available. The DSAWG decision dates for the PPS are published in the CAL.

Registration

DoD Instruction 8551.1 requires that all existing, new, and planned DoD information systems visible to DoD-managed network components must be registered in a PPS registry maintained by DISA. This process ensures that all necessary ports remain open as long as a DoD information system has PPS that cross enclave boundaries into the DoD network space.

Each DoD component has a point of contact (POC) who is authorized to register information systems. A list of the POCs is available at NetDefense Joint Task Force-Global Network Operations [6]. Once registration is completed, the process will automatically notify DoD network administrators to open ports and protocols as required on appropriate DoD routers and firewalls.

If the registration is for a new PPS configuration in a newly developed or newly acquired DoD information system, or for a modification of an existing DoD information system, the PPS will be temporarily opened, as required, until the DSAWG has accepted the risk of the PPS within the DoD network space. The assessment and assignment of categories to PPS takes approximately three months. Therefore, it is important to register information systems as soon as possible in the development or acquisition process to ensure availability of the intended PPS.

Benefits

Information system managers, architects, and developers in today's software development environment are rapidly being led to the realization that security is not an add-on feature but must be identified as a core requirement from the beginning. This concept is also referred to as *baked in security*. The PPSMP registration process enables DoD components to identify and eliminate poor business and security practices (e.g., unencrypted remote management of routers and firewalls from the Internet). This section identifies the benefits of the PPSMP to PMs, system engineers, software developers, designated approving authorities, and network operations staff.

Many protocols and network-based services were designed by people who were not concerned with malicious behavior. As a result, a computer that uses these protocols and services is exposed to attack. However, many of these risky protocols and services are useful, and a few are even necessary.

One way of dealing with the risky protocol problem is to limit the population that can interact using a particular protocol or service. In practice, this is often done by limiting the use of a risky protocol to the local workgroup, or local area network. A firewall or filtering router blocks the protocol at the group boundary, thereby shielding members of the

group from the people on the *outside* who might attack machines on the inside that have the protocol enabled. Although in this case a firewall is an enabler of a capability, it can also cause interoperability problems if people or applications on either side of the boundary must use that risky protocol for essential communication across the boundary.

The ports and protocols process is the method the DoD uses to determine the *riskiness* of particular protocols and services, and then balances that risk against the operational utility of the protocol with guidance on when and how to use the protocol and service.

Program Managers

By providing PMs with the list of approved PPS, the DoD allows PMs to target the acquisition of systems and system components that must meet interoperability goals and information assurance goals. DoD PMs can request that the PPSMP process review considers using PPS prior to making costly implementation decisions. The DoD PPSMP was developed with the understanding that DoD PMs are continually focused upon multiple challenges, as follows:

- Provide and refine products and services in conjunction with the mission element needs statement, required operational capability, major automated information system review council, and defense information technology security certification and accreditation process processes.
- Meet the schedule.
- Execute fiscal responsibility.
- Minimize fielding costs.
- Minimize software maintenance costs.
- Reduce time to accredit and time to field.

PMs will no longer need to develop multiple system baselines to comply with different component firewall policies. The PPSMP (DoD Instruction 8551.1, paragraph 4.8) requires network security administrators to open enclave boundaries for properly registered AISs using recommended PPS. Early adopters of the PPSMP are Health Affairs and the High Performance Computing Office.

System Engineers

The PPSMP provides the ability to choose compatible products early in the system development cycle and supports a standardized architecture, which reduces configuration management issues with network connections. The PPSMP also simplifies the accreditation process by encouraging standard implementation and consistent reporting methods. It reduces rework costs for system fielding due to PPS cross-component conflicts.

Software Developers

Software developers will be able to use the standard implementation configurations provided in the CAL. The configurations identified in the CAL provide the target architectures that have been vetted by the DSAWG.

Designated Approving Authorities

The PPSMP supports the designated approving authorities (DAAs) by identifying the technical risk of using specific PPS. This identification enables the DAAs to perform informed risk assumption evaluations and decisions. The PPSMP also reduces DAA staff evaluation time for registered AISs and PPS, which use standard configurations and are fielded in full compliance with DISA security technical implementation guides.

"The PPSMP provides the ability to choose compatible products early in the system development cycle and supports a standardized architecture, which reduces configuration management issues with network connections."

Network Operations Staff

The Network Operations staff includes both network operators and system administrators. The PPSMP supports the Network Operations staff by providing standard implementation configurations, standardizing router configurations with predefined rule sets to be used as required to meet operational requirements, and reducing hostile/unintended traffic.

Network operators are responsible for the operations and maintenance of major segments of managed networks. Network Operations customers include the United States Strategic Command Joint Task Force-Global Network Operations (JTF-GNO), DISA operations, JTF-GNO Global Network Center, Global Network Support Center, Theater Network Centers, NetDefense, and Computer Emergency Response Teams and Network Operations Centers of the DoD components.

System administrators are responsible for the installation and maintenance of information systems, providing effective information system utilization, and ensuring the use of adequate security parameters and sound implementation of established information assurance policy and procedures.

Benefits realized by the Network Operations staff include advance notice of specific vulnerabilities, potential attack vectors known before exploits exist (e.g., Blaster, Slammer), and aid in the immediate impact analysis of potential port closures during attack/protection decisions. Other benefits include the standardization of router configurations with predefined rule sets to be used as required and the reduction of hostile/unintended traffic.◆

References

1. Department of Defense. "Ports, Protocols, and Services Management (PPSM)." DoD Instruction 8551.1. Washington: DoD, 13 Aug. 2004 <www.dtic.mil/whs/directives/corres/html/85511.htm>.
2. The Internet Engineering Task Force. Internet Society. 8 Feb. 2005 <www.ietf.org>.
3. Internet Assigned Numbers Authority. 12 July 2004. Internet Society. 8 Feb. 2005 <www.iana.org>.
4. Defense Information Systems Agency. Information Assurance Support Environment. DISA. 8 Feb. 2005 <http://iase.disa.mil>.
5. Defense Information Systems Agency. "PPS Assurance Category Assignments List" Release 2.0. Ports and Protocols. Feb. 2005 <http://iase.disa. mil/ports/index.html>.
6. Department of Defense. "Net Defense (DoD-CERT) Branch of JTF-GNO at JTF-GNO." DoD Ports and Protocols Program. 8 Feb. 2005 <www.cert.mil/portsandprotocols>.

Note

1. See <http://ipv6.disa.mil>, <www.ipv6.org>, and <www.ipv6forum.com>.

About the Authors

David R. Basel is the project manager for the Department of Defense (DoD) Ports and Protocol Management Project. He is a member of the Defense Information Systems Agency (DISA) Global Information Grid Combat Support Directorate's Center for Network Services. Basel has been with DISA for 13 years. He has held several first-line supervisory and chief engineer positions, including division chief for Communications Network Security, chief engineer for DoD Share Data Environment, and deputy program manager for the DoD Ada Program. Basel has a Bachelor of Science in computer science from Bowling Green State University.

Center for Network Services
GIG Combat Support Directorate
DISA
P.O. Box 4502
Arlington, VA 22204-4502
Phone: (703) 882-1553
Fax: (703) 882-3336
E-mail: disapao@disa.mil

Dana Foat is a computer systems analyst assigned to the Defense-Wide Information Assurance Program Office, Office of the Assistant Secretary of Defense for Networks and Information Integration. Prior to his current assignment, he held various network security positions with the Information Assurance Directorate at the National Security Agency. Foat is a Certified Information Systems Security Professional. He has a Bachelor of Science in mathematics from Valparaiso University, a Master of Science in computer science from Johns Hopkins University, and a Master of Business Administration from the University of Pittsburgh.

Defense-Wide Information
Assurance Program Office
1215 S Clark ST
STE 1101
Arlington, VA 22202-4302
Phone: (703) 602-9974
Fax: (703) 602-7209
E-mail: dana.foat@osd.mil

Cragin Shelton is a senior Information Security engineer with The MITRE Corporation. He has been involved with management of Department of Defense (DoD) information systems and networks since 1984. Since retiring from the U.S. Air Force in 1997, he has concentrated on information security and information assurance aspects of DoD systems and networks. Shelton is a Certified Information Systems Security Professional. He has a Bachelor of Science in chemistry from Centenary College of Louisiana and a Master of Science in Systems Management (information systems) from the University of Southern California.

The MITRE Corporation
7515 Colshire DR
MS T320
McLean, VA 22102
Phone: (703) 883-5836
Fax: (703) 883-1245
E-mail: cshelton@mitre.org

Transformational Vulnerability Management Through Standards

Robert A. Martin
MITRE Corporation

The Department of Defense's (DoD) new enterprise licenses for vulnerability assessment and remediation tools [1, 2] require using capabilities that conform to both the Common Vulnerabilities and Exposures Initiative's [3] and the Open Vulnerability and Assessment Language Initiative's [4] standards efforts, as does a new Air Force enterprise-wide software agreement with Microsoft [5]. These contracting activities are part of a larger transformation of the DoD's management and measurement of the information assurance posture of their network-enabled systems with respect to vulnerabilities, configuration settings, and policy compliance. In combination with procedural changes, the adoption of these [6] and other standards such as the National Security Agency's Extensible Markup Language Configuration Checklist Data Format [7], are making it possible to radically improve the accuracy and timeliness of the DoD's remediation and measurement activities, which are critical to ensuring the network and systems integrity of their network-centric warfare capabilities.

Tuesday, 19 April 2005
Track 1: 3:55 – 4:40 p.m.
Ballroom A

The basic process for addressing unexpected security-relevant flaws in any commercial or open source software used in any organization, including the Department of Defense (DoD), starts with the discovery of a security-relevant flaw in that software. The discoverer could be the software creator, an outside researcher, or a software user. The next step is usually for the software creator to be informed of the potential security-relevant flaw to start evaluating it and looking for potential resolutions.

Eventually, a fix and possible workaround to the flaw(s), if it turns out to be real, are released to software customers. This is usually done via a security advisory or bulletin from the software creator and/or by the researcher who discovered the flaw. Subsequently, the community of security tool developers that checks for security flaws in deployed software starts the task of figuring out how to check for this new public security flaw and its fixes.

For most of these developers, their only information to start with is the narrative from the security advisory or bulletin. In short order, most security assessment tool developers will update their tools to look for and report systems status regarding this new security-relevant flaw. Exactly how each tool checks for the flaw and its possible resolution is usually not known to the tool users.

DoD's Current Flaw Management and Measurement Process

In the DoD, there is keen interest in ensuring that critical security-relevant flaws are sought out and addressed in a timely manner. Not all flaws that are discovered and made public will be relevant to the DoD. Only those that involve the specific platforms, operating systems, and applications in use in the DoD are of interest.

> *"... most remediation tools treat the results of assessment tools as good suggestions of where to start looking for flaws, and then end up doing their own assessment before making recommendations on remediation approaches."*

The DoD process of identifying which publicly known flaws need to be addressed and the timeframe for addressing them results in one of three notifications: Information Assurance Vulnerability Alerts (IAVAs), Information Assurance Vulnerability Bulletins (IAVBs), and Information Assurance Vulnerability Technical Advisories (IATAs) [8, 9]. Depending on the potential impact of the flaw, it will be included in one of these three notifications – unless the impact is thought to be insignificant.

DoD organizations are responsible for addressing the flaws discussed in these different notifications and for recording their progress and completion in resolving the flaws. Collectively, this is referred to as the *IAVA process*. A new flaw that must be assessed, reported upon, and remediated can be referred to as a new *IAVA requirement*.

Today, that process is very dependent on manual reporting methods, as illustrated in Figure 1, which starts with a known *compliant* system that has addressed all known flaws. The figure shows how the discovery of a new flaw proceeds to the assessment for that flaw, followed by the reporting of the status with respect to that flaw, and the remediation of the flaw, with the subsequent return to a known compliant state.

Quick, Complete, and Dependable Knowledge

There are many opportunities for improving the original IAVA process and the information sources upon which it relies. Some of the most striking opportunities are improving the quality of the information contained in the original announcements about new flaws; improving the accuracy, completeness, and timeliness of security tool vendor's incorporation of tests for new flaws; minimizing the dependence on manual reporting within the enterprise; overcoming the difficulty in combining reports and findings from various tools

due to differences in their format, test criteria, and test methods; and eliminating the need for reassessment as part of the remediation process. The opportunity for improvements in all of these areas comes from the lack of standardization in the vulnerability management arena.

Standard Machine-Readable Flaw Definitions

Why do we tolerate having each security tool developer re-create the knowledge of how to identify a new flaw? We all know the organization that tells everyone about the flaw has much more knowledge about it. Usually, they have been studying the flaw and its possible remediation methods for some time while they prepare to make the flaw public. If the flaw discoverer could pass his or her knowledge along to the tool developers in a quick and precise way, we all would benefit, especially if it was done in a non-exploitive manner. Most advisory/bulletin writers try to explain how to determine if you have the flaw they are writing about. However, they do it in narrative English and without any consistency. The Open Vulnerability and Assessment Language (OVAL) Initiative [4] is an extensible markup language-based language standard that is specifically designed to address this issue.

Different Criteria

Why do we put up with different tools using alternative methods to determine whether a particular flaw exists on one of our systems? Currently, one assessment tool might examine the banner reply from a networked service to determine whether the flawed software is installed. Another tool may try an exploit over the network to see if the flaw exists. A third tool might authenticate itself to the system to gain access to the file system-level information, determining whether the flawed software version is installed and then checking for the appropriate patch or service pack. Finally, a fourth tool may conduct these system-level checks, and then also check whether other remediation approaches, like changing the ownership of the flawed software to root (which makes it unavailable for general users to run), could make the flaw unexploitable.

If an organization has different tools using different testing methods – with most of the test criteria being hidden – it is easy to see that combining the results of tools may not be straightforward. Additionally, not knowing how a tool is checking for a flaw can make it very difficult to determine whether systems are safe or exploitable. With such a large variety of test methods and results, most remediation tools treat the results of assessment tools as good suggestions of where to start looking for flaws, and then end up doing their own assessment before making recommendations on remediation approaches. Since time is a critical factor, why should we be doing assessments for the same flaws more than once?

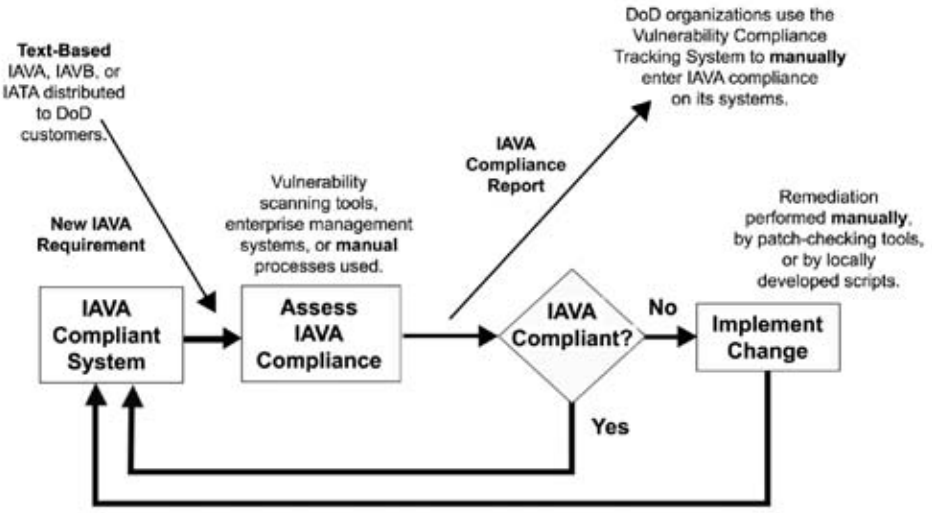


Figure 1: *IAVA Process*

The OVAL Initiative is designed to address these issues. Additionally, most network assessment-based tools are adding capabilities to allow for authenticated access to systems so they can produce more definitive findings as to whether a particular flaw exists on a system. This trend allows network tools to use OVAL test definitions for these new checks.

Combining and Communicating Vulnerability Assessment Information

What would be the impact from having a choice of different assessment tools that were all using known testing criteria and each provided standardized results? Assuming that these results contained the minimum necessary information to allow an organization to combine the findings of different tools to create an organizational status report, we could stop trying to use a single, all-encompassing tool and instead select appropriate tools based on what they do well. For instance, one tool might do well on network components like routers and switches, another tool might cover security appliances, another tool may address Windows-based standard applications, another Solaris, and so on.

With standard results formats and structures, we could get the enterprise insight we need without giving up the power of specialization. The OVAL Initiative's Result Schema is specifically aimed at addressing this. Additionally, with the right type of information being passed along we could eliminate some portion of the redundant assessment work that remediation tools are forced to undertake today.

DoD's Future Flaw Management and Measurement Process

By utilizing the Common Vulnerabilities and Exposures (CVE) Initiative [3], OVAL, and eXtensible Markup Language (XML) Configuration Checklist Data Format (XCCDF) standards [7], the DoD will be able to transform the IAVA process into one that is predominantly based on machine-to-machine information flows that will improve the accuracy, timeliness, and manpower needed to address the flaws that are found in software.

Figure 2 (see page 14) illustrates the revised IAVA process. *New IAVA requirements* include OVAL definitions on how to identify the new issue. Assessment tools are capable of using the OVAL definitions; they report their findings per the OVAL results XML standard. These same standard-based results are fed into the reporting process and the remediation process. Various procurements have started requiring

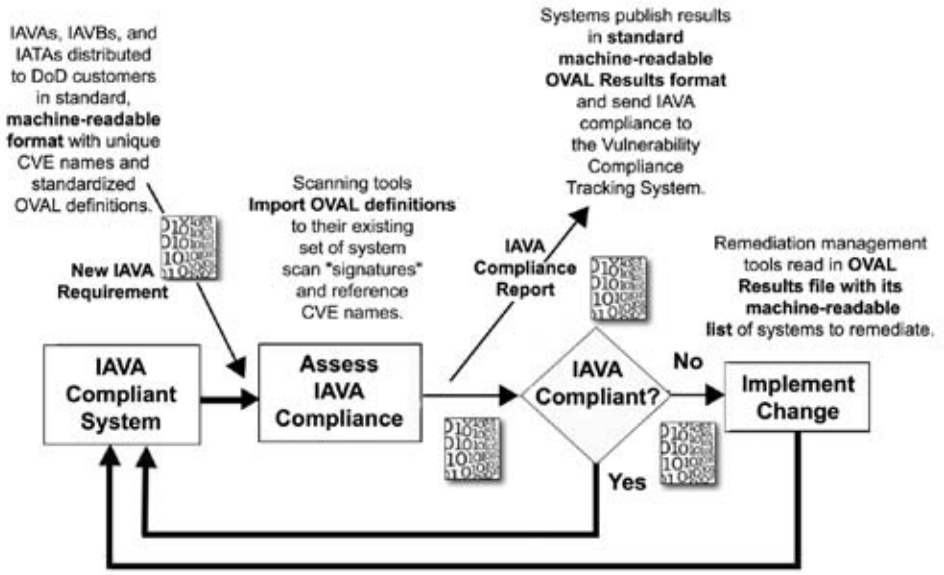


Figure 2: *Standard-Based IAVA Process*

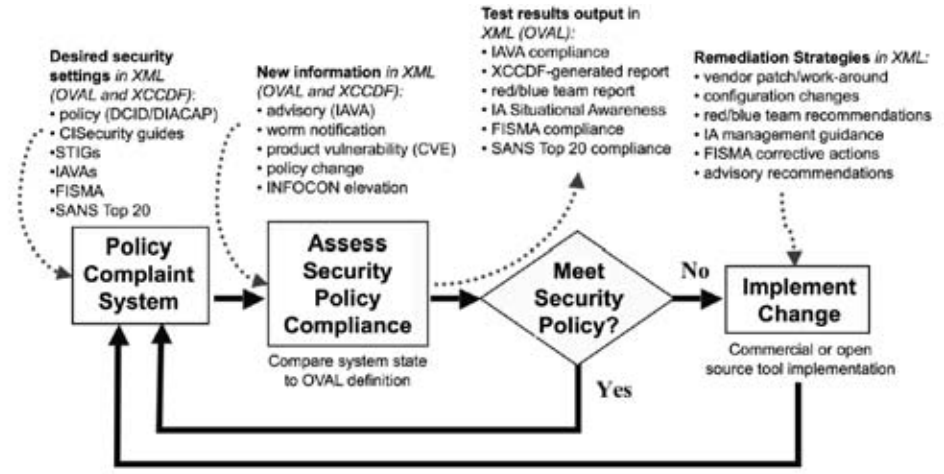
support for the standards that will enable the transition to this new IAVA process. Work in transforming current checklists and checking guidelines into these standards is also under way, which will set the stage for the formal process to be changed.

Dealing With More Than Vulnerabilities

The DoD, like many organizations, has established criteria for how its operating systems and standard applications are configured. These criteria are usually different from the way the software suppliers ship the software from their distribution facility. The DoD, through the work of the Defense Information Systems Agency (DISA), the National Security Agency (NSA), the Center for Internet Security (CISecurity), and several vendors, has reached a consensus over the past few years on how operating systems and applications can be *locked down* to safer configurations. These settings can be checked by the free tools that CISecurity provides, but in the near future these configuration checks will be available as machine-readable XML policy documents that use a combination of NSA's XCCDF and OVAL [10]. Additionally, the DoD's Security Technical Implementation Guidelines' configuration guidance [9] can be expressed as XML documents using XCCDF and OVAL, which would make both of these collections of policy tests on configuration settings usable within commercial and open source security tools that are able to import the XML test definitions.

Similarly, the testable portions of other policy efforts can be expressed as a combination of XCCDF and OVAL XML. Doing so opens the door to increased automation and improved fidelity in enterprise status reporting and management with respect to these efforts. The Director of Central Intelligence Directive (DCID) 6/3 [11], Defense Information Assurance Certification and Accreditation Process (DIACAP), Federal Information Security Management Act (FISMA) [12], and The SANS [SysAdmin, Audit, Network, Security] Institute's Top 20 [13] can all benefit from the clarity and automation that expressing their goals in machine-readable standardized languages provides. It can probably also significantly change the amount of time and labor that organizations dedicate to reporting and managing these efforts, versus adjusting its systems to comply with them. Figure 3 illustrates how adoption of these types of standards could look.

Figure 3: *A Standard-Based Security Management Process*



Patch Applicability Testing

The same types of information that are used to test for flawed software, misconfigurations, and adherence to stated policies can be used to check whether a particular patch can be applied to a system. The OVAL language includes a patch definition type that will support testing whether the prerequisites for a patch are fulfilled, allowing an assessment tool to determine whether a particular patch can be applied to a system. Collectively, the CVE, OVAL, and XCCDF standards describe a collection of interoperable functionality that will streamline the way security assessment and management are applied in the enterprise, opening the door for more interoperable and composable tool support as shown in Figure 4.

Conclusion

The DoD's new vulnerability and configuration standardization efforts are focused on the elimination or minimization of manual and non-automated aspects of these areas. The DoD is moving to its new process by requiring the inclusion of CVE names and standardized OVAL XML vulnerability and configuration tests in software supplier's alerts and advisories, and by acquiring tools that can import new and future OVAL XML test definitions and export their findings as standardized OVAL XML results.

By also obtaining capabilities that can import the OVAL XML results for remediation, organizational status reporting, and generating certification and accreditation reports, the DoD will have created a focused, efficient, timely, and effective enterprise incident management and remediation process by adopting information security products,

services, and methodologies that support the CVE naming standard and use OVAL test definitions and results schemas. By also adopting the XCCDF standard, the DoD will be able to take the improvements in these areas on to a fuller set of policy and configuration management arenas.

Collectively these changes will dramatically improve the insight and oversight of the security and integrity of the systems and networks underlying tomorrow's network-centric warfare capabilities.◆

Acknowledgments

The summary work contained in this article is based on the efforts of a large number of individuals; a special thanks is given for the contributions of Julie Connolly.

References

1. Defense Information Systems Agency. "DISA IASSURE, Contract 2004, Task Order 232, Statement of Work for eEye Digital Security's Retina." Washington: DISA, 3 June 2004 <www.ditco.disa.mil/public/discms/IASSURE/00232_00.doc>.
2. Defense Information Systems Agency. "DISA IASSURE, Contract 2004, Task Order 254, Statement of Work for Citadel's Hercules." Washington: DISA, 24 Sept. 2004 <www.ditco.disa.mil/public/discms/IASSURE/00254_00.doc>.
3. The MITRE Corporation. "The Common Vulnerabilities and Exposures (CVE) Initiative." Bedford, MA: MITRE Corporation <http://cve.mitre.org>.
4. The MITRE Corporation. "The Open Vulnerability and Assessment Language (OVAL) Initiative." Bedford, MA: MITRE Corporation <http://oval.mitre.org>.
5. Fisher, Dennis. "Microsoft, Dell Snag Air Force Deal." eWeek 29 Nov. 2004 <www.eweek.com/article2/0,1759,1731420,00.asp>.
6. Department of Defense. "Information Assurance (IA) Implementation." DoD Instruction 8500.2, Section VIVM-1, Vulnerability Management, Mission Assurance Categories I, II, and III. Washington: DoD, 6 Feb. 2003: 64, 75, 84 <www.dtic.mil/whs/directives/corres/html/85002.htm>.
7. Ziring, Neal. "Specification for the Extensible Configuration Checklist Description Format (XCCDF)." Ed. John Wack. Washington: National Institute of Standards and Technology, Jan. 2005 <http://csrc.nist.gov/checklists/docs/xccdf-spec-1.0.pdf>.
8. Defense Information Systems Agency. DISA IAVA Process Handbook. Ver. 2.1. Washington, DC: DISA, 11 June 2002 <www.tricare.osd.mil/contracting/healthcare/solicitations/TDP/0000/00_Attachment_16.pdf>.
9. Joint Task Force – Global Network Operations/NetDefense [DoD-CERT]. 25 Jan. 2005 <www.cert.mil>.
10. Beale, Jay. "Big O for Testing." Information Security Dec. 2004 <http://infosecuritymag.techtarget.com/ss/0,295796,sid6_iss526_art1075,00.html>.
11. Federation of American Scientists. "Director of Central Intelligence Directives 6/3 Protecting Sensitive Compartmented Information Within Information Systems." Washington: FAS, 5 June 1999 <www.fas.org/irp/offdocs/DCID_6-3_20Policy.htm>.
12. National Institute of Standards and Technology. "Federal Information Security Management Act of 2002. Title III – Information Security." E-Government Act (Public Law 107-347), Dec. 2002 <http://csrc.nist.gov/policies/FISMA-final.pdf>.
13. The SANS Institute. "The Twenty Most Critical Internet Security Vulnerabilities - The Experts Consensus." Ver. 5.0. Bethesda, MD: The SANS Institute, 8 Oct. 2004 <www.sans.org/top20>.

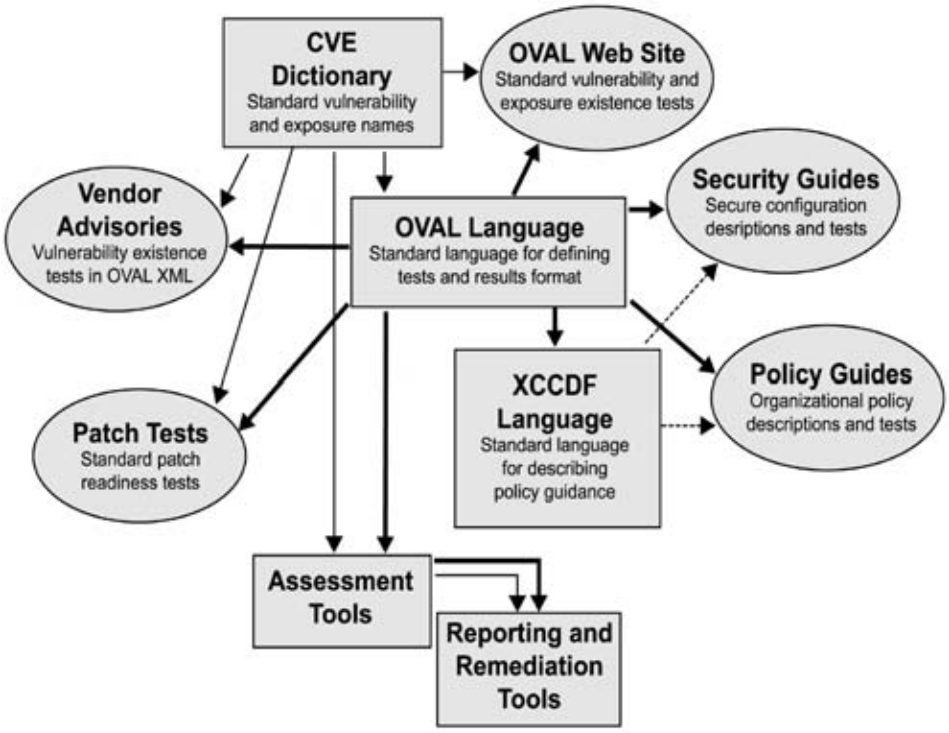


Figure 4: *Standard-Enabled Information Security Automation*

About the Author

Robert A. Martin is a principal engineer in The MITRE Corporation's Information Technologies Directorate. For the past five years, Martin's efforts have been focused on the interplay of risk management, cyber security standards, critical infrastructure protection, and the use of software-based technologies and services. He is a member of the Association for Computing Machinery, the Armed Forces Communications and Electronics Association, the Institute of Electrical and Electronics Engineers (IEEE), and the IEEE Computer Society. Martin has a bachelor's degree and a master's degree in electrical engineering from Rensselaer Polytechnic Institute, and a Master of Business Administration from Babson College.

MITRE Corporation
202 Burlington RD
Bedford, MA 01730-1420
Phone: (781) 271-3001
Fax: (781) 271-8500
E-mail: ramartin@mitre.org

Extending Agile Methods: A Distributed Project and Organizational Improvement Perspective

Paul E. McMahon
PEM Systems

Tuesday, 19 April 2005
Track 7: 4:50 – 5:35 p.m.
Room 251 A-C

It has been argued that agile methods only work for small, collocated, self-directed teams that include on-site customers. But what if your customer cannot be on-site full-time, or your development team is distributed around the world, or your developers lack self-directed team skills? Does this mean you cannot take advantage of agile methods? This article presents a case for using key agile practices along with recommended extensions on a broader range of projects, including large and physically distributed efforts. The article motivates the use of agile methods by exposing common myths and providing information that can help managers and customers facilitate practical agility within their organizations.

Many organizations are looking seriously at agile methods to see if there are benefits to be gained across a broader range of projects. Unfortunately, today these methods are often misunderstood and misapplied. To aid understanding, let us start with some fundamentals and misperceptions.

The Agile Manifesto, which was put together by the founders of many of the most popular agile methods, contains four value statements:

- We value individuals and interactions over processes and tools.
- We value working software over documentation.
- We value customer collaboration over contract negotiation.
- We value responding to change over following a plan. [1]

A key agile value is customer collaboration. This value can be observed through User Stories [2] that are employed by eXtreme Programming (XP) for requirements. User Stories are intentionally high-level with details worked out collaboratively and informally between developer and customer. Nevertheless, I have heard a customer say, "I do not see how agile can help, because I need all my requirements." The belief that agile means that customers must live without all requirements is a misperception.

Another common agile value is working software demonstrated to customers often through short development iterations. The motivation for this value is the belief that a better way to ensure customer needs are met is through working software rather than through formal written words.

In response, I have heard customers and contractors say the following:

- "The lack of up-front planning and requirements on agile projects leads to chaos."
- "Short iterations do not work on large projects because there isn't time to get the design done."
- "Short iterations on complex projects lead to team burnout."

To motivate and provide greater insight, let us now discuss six agile myths.

Myth 1: Agile Methods Do Not Include Plans and Requirements

Many who claim to be agile are in fact using a code-and-fix approach. Agile is not code and fix. It is both incremental and iterative; however, its iterative aspect is often misunderstood. Iterative means that inside each increment there are smaller cycles of development occurring (each usually from four to 12 weeks).

The important point often missed is that each iteration is not just code, but includes plans, requirements, design, code, and test. Those familiar primarily with traditional development methods used on large projects often do not understand how this is possible. The key to agile lies in the fact that the activities, their sequence, and the resultant agile artifacts are not traditional.

If you are a customer or manager familiar only with traditional development methods and you want to take advantage of agile methods, then you will want to know how to distinguish agility from code and fix. Understanding a few practical rules can help.

Practical Rule 1: Every Agile Iteration Is Planned and Measured Regardless of Iteration Length [3]

Agile projects use a two-tier approach to plans: a long-term, coarse-grained plan, and a short-term, fine-grained plan. [4]. On agile projects, planning occurs continually to ensure the team is always focused on *the most important things now*. I was asked by one client, "How can I tell if my team is really doing agile planning, or just reacting to the next fire?" Asking your team the following two questions can help.

- **Question 1:** "How do you determine the most important things?" After you ask this question, listen for the word *risk*. If your team is really doing agile planning, you should hear how the risks perceived by both the development team and the customer are being handled collaboratively. Teams that are reactive often do not take time to collaborate.
- **Question 2:** "How do you reflect the results of your continual planning?" If you hear, "We are agile so we do not document our plans," then your team is not agile. Contrary to what you might have heard, agile teams do document their plans, but the resultant planning artifacts look different. Examples of agile planning artifacts are allocation of user stories to iterations, and task sign-up sheets.

Myth 2: Agile Methods Do Not Allow Requirements Control

Agile methods do not guarantee requirements control, but they do allow it. One way requirements can be controlled with agile methods is to use two levels of requirements. The first level is the high-level User Stories that scope the *complete* project. This level includes features that have not yet been fully analyzed. In Scrum this potential work is placed on what is referred to as the Product Backlog.

The second level of requirements is developed collaboratively with the customer and establishes in greater detail and clarity the work and priorities for the next iteration. In Scrum this work is referred to as the Sprint Backlog.

Not long ago, the company president of one of my agile clients was having a beer with one of his customers. As the

customer was explaining a desired feature, the president looked the customer in the eye and said, "We can do that." A few weeks later the customer was observing a demonstration of the current iteration software and became upset because he did not see the desired feature.

It is worth noting that the customer's desired feature referred to in this story may or may not be within the scope of the current requirements. To ensure requirements are controlled, I recommend that customer requests such as these be first placed on the Product Backlog. Once this potential work is analyzed, clarified, and approved, it may then be placed on the Sprint Backlog. The Sprint Backlog is fixed at the start of each iteration.

The point of the story is simple. Customer collaboration does not mean the contractor must implement everything the customer asks for. With agile, there can still be out-of-scope requests. The agreed-to detailed requirements are established collaboratively by the customer and contractor for each iteration.

Practical Rule 2: The Work for Each Agile Iteration Is Fixed At the Start of Each Iteration [3]

While these practical rules may seem obvious, do not dismiss them lightly. With agile methods we have few prescriptive rules – and this can aid team productivity – but only if those few rules are consistently followed to keep the agile team from falling into chaos.

It is worth noting here that Alistair Cockburn pointed out to me that while this type of rule is a good starting place, some advanced agile groups allow more dynamic changes to the content of an iteration. This has been referred to as Dynamic Scrum [5].

Myth 3: The Schedule Never Slips With Agile Methods

While it is true that with agile methods we keep each iteration a fixed length, this does not mean the schedule never slips. I recommend planning with a *buffer* iteration at the end that starts with no stories allocated to it. This gives management the time to take action by moving incomplete work to the buffer. But when the buffer overflows, you must add another iteration and own up to a schedule slip. This addresses the customer concern of not getting all their requirements.

Myth 4: Agile Methods Are Only for Programmers

Agile methods do help programmers, but the benefits extend far beyond code. At the 2004 Systems and Software Technology Conference (SSTC), the U.S. Government's Top 5 Quality Software Projects for 2003 were awarded. At this presentation, Linda Crabtree, a Development Group lead on the Patriot Excalibur Project (one of the award winners), talked about how her project was having trouble meeting schedules and keeping the customer satisfied [6]. After her team adopted XP, they began to hit their schedules more consistently, and customer satisfaction increased.

To understand how an agile method can have such a dramatic effect requires a deeper understanding of the first agile value. Traditionally, we plan new projects based on similar past projects. Schedules are often developed assuming personnel with skills similar to those on past projects will be assigned.

> *"With agile, instead of predicting schedule performance based on results from a different project with different people, we plan our team velocity continuously based on the actual team's current performance."*

Agile is different. With agile, instead of *predicting* schedule performance based on results from a different project with different people, we plan our team velocity continuously based on the actual team's current performance. With agile methods, this is possible because of the short iterations that provide actual team results early and often. Keeping each iteration a fixed length, referred to as time-boxing, is key to accurate velocity measurement.

Patriot Excalibur is not the only project reporting positive project management results using agile methods. I and other conference attendees heard a similar report at the 2004 SSTC from David Webb, a technical program manager for the Software Division of Hill Air Force Base [7].

Through agile methods, we are learning that people are not commodities. That is, you cannot just pull one *individual* off a project and plug another in and expect to get the same results. This fact does not change as projects increase in complexity or size.

It is important to note here that key to both of these reported successes was not only early visibility of actual team velocity, but also a customer willing to collaborate, as was reported by both Crabtree and Webb.

Myth 5: You Get No Design With Agile Methods

I heard a manager in a large company say, "We tried Scrum [8], but I wouldn't recommend it because we ended up with no design." Some mistakenly believe there is not time to get design done when using agile methods.

What is different with agile is *when* the design is done. With agile, we do not limit design to a fixed time slot within a fixed phase. In fact, we encourage deferring design details – not skipping them. What is often missed by those comparing agile to traditional methods is the extensive cost of design rework that frequently occurs during integration with traditional methods.

Agile encourages doing design at the optimum time (e.g. when data is available and high-level requirements have been clarified) to minimize rework and thereby maximize overall team velocity.

When you hear someone say, "We ended up with no design," what they often mean is that they ended up with no documentation of the design. This leads to the next myth.

Myth 6: Agile Methods Do Not Allow Documentation

Note that the second Agile Manifesto value is a relative statement. It is a myth that agile methods do not allow documentation. However, most agile methods are silent on this subject, leaving documentation decisions up to the project [4]. This includes both deliverable documentation and *process* documentation (e.g., action items, meeting minutes). This subject has also been referred to as the *ceremony* of the project [9].

When making documentation decisions, recognize that it is not a matter of being agile or not agile. There exists varying levels of agility, but when planning your project's ceremony, be aware that short iterations and high ceremony may place your project schedule at high risk.

Now let us turn our attention to extending agile methods to a broader

range of projects, including physically distributed efforts.

Extending Agile to Large and Distributed Projects

After working on three failed multi-organization physically distributed projects in the late 1990s, I spent a year researching similar projects looking for causes and solutions [10]. At the heart of the difficulties, I found communication breakdown leading directly to increased project integration risk. A colleague brought to my attention that the solutions I was advocating had similarities to the agile movement. Ironically, many of the agile experts were saying do not try these practices on large and distributed efforts.

If you attempt agile methods *out of the box* on large and distributed projects, you are likely to fail. This is because these methods require extensions to work on more complex projects. As I discuss these extensions, I will also explain some of the wrong ways to extend agile methods.

For example, recognizing the need to address the integration risk some distributed projects in the past have employed *heavyweight* architectures (e.g., formal front-end design reviews, extensive presentations, lengthy written documents). These same teams often failed to deliver acceptable working software. You can fail by collaborating too much, too little, or in the wrong places.

Recommended Extension 1: Agile Architecture

I recommend employing *agile architecture*, which involves first setting up an *agile architecture team*. Think of agile architecture like agile requirements – there are two levels. At the first level, a *small* strategically selected team rapidly develops a high-level agile architecture and documents the results. The goal of the agile architecture is to address the complete project scope and provide a minimum set of architecture compliance rules (e.g., hardware platform requirements, minimum interfacing rules).

The resultant product of the first level is a *thin* architecture document that includes a simple, high-level (but complete) diagram showing the major system components. This document also includes high-level assumptions and a brief description of each component.

Agile architecture is similar to User Stories in that it represents a commitment to talking to solve detailed architecture issues collaboratively with the agile teams during each iteration. Key to keeping the architecture agile is communicating the simple high-level diagram and the minimum compliance rules to each agile team. At the second level, the agile architecture team focuses on the high-risk areas for each iteration. Cockburn has referred to these areas as the big rocks [11].

Each specific architecture solution is documented by the agile architecture team through a *lightweight position paper*. These position papers may be maintained separately, or appended to the thin architecture document. The architecture grows over time as the solutions accumulate. Crucial to agile architecture scalability is the strategically selected architects who use their experience to determine where the architecture is best kept simple, and where the big rocks lie.

The use of agile architecture has been proven to be successful on past large distributed projects that have employed hybrid agile methods [12]. XP refers to *metaphor* [13] to address architecture, but metaphor is too weak for many complex efforts, especially when the team members do not reside at the same location.

Scaling Up Agile Teams: The Wrong Way

While communication breakdown plagues many large distributed efforts, improved team communication rests at the core of agile team success. Key to this improved team communication is the self-directed daily stand-up meeting.

I heard a manager on a large project that was attempting to scale up an agile method say, "We cannot afford a manager for every six to eight people." His project was holding daily stand-up meetings with more than 30 people. Unfortunately, when you scale up daily stand-up meetings this way the meetings tend to lose their self-directed quality.

Scaling up agile teams and maintaining improved team communication can be tricky. I recommend keeping the agile team's stand-up meetings to a reasonable size (less than 10), even if the full project has 500 or more people. Critical to the success of the daily stand-up meeting is the individual. Each must be heard. When stand-up meetings get too large, leaders start directing rather than listening and agile benefits are lost.

Some misunderstand the role of the agile team lead (e.g., ScrumMaster, XP Coach/Tracker) [8, 13]. A primary responsibility of the lead is to listen and then do everything possible to remove obstacles that are hindering the team. Too often this critical role is understaffed, especially when organizations attempt to scale up agile methods inappropriately. When the lead is not available to work on issues daily, communication breaks down and the team loses velocity – along with the key benefits of agile methods.

Another pitfall that has been observed when scaling up agile methods on large projects is a *stovepipe-mentality* among the individual agile teams. In other words, when a large project is partitioned into many small agile teams, improved communication inside each team may occur as expected, but at the expense of reduced communication across the full project.

Recommended Extension 2: Use Super Leads to Scale Up Agile the Right Way

One way to scale up agile methods and avoid the pitfalls discussed is to use *super leads*. Each super lead may oversee between three and five agile teams. It is important to understand that the super lead is not the agile team lead. Super leads may or may not attend daily stand-up meetings but if they do, they do not speak. The super lead's role is primarily a mentoring role for less experienced agile team leads, and a communication role.

Super leads review agile team plans and metrics (e.g., velocity charts) providing feedback directly to the agile team lead – not the team members. This approach can help address the observed lack of self-directed team skills on many large and distributed efforts. It is important that the super leads do not direct the agile teams; otherwise, we risk losing a primary benefit of agile methods – improved visibility of real status early and often.

The super lead oversees multiple agile teams with an eye on cross-team communication, ensuring the agile architecture team is engaged at the right time. This can reduce the stovepipe mentality.

I heard one agile team lead say, "The short iterations are causing my team to get burned out." This comment might be a warning sign to a super lead that help is needed. A key to successful implementation of agile methods is a self-directed team that can measure its own velocity, project its future velocity, and communicate the results up the chain. If an agile team finds itself working 80-hour weeks, this is a sign that they may not be accurately measuring their velocity, or they are not controlling their work tasks, or they are not communicating effectively the real status up the chain. A more experienced agile team lead will be able to spot these signs and help get the team back on course.

Even if they are in short supply, using your best leaders across multiple teams

can help facilitate communication on large and distributed projects, as well as help jumpstart agility on a broader scale in your organization.

It is worth noting that the super leads may meet periodically as a team. A similar concept (Scrum of Scrums) has been discussed by Ken Schwaber on scaling of agile methods [8].

Recommended Extension 3: Use Super Leads as Customer Proxies and to Aid Customer Communication

Often, especially on larger projects, customers cannot be on-site full-time. A primary motivator for having an on-site customer is to answer questions quickly so the agile team does not lose velocity. Sometimes customer proxies (e.g., subject matter expert) can serve this purpose, especially on large and distributed projects. When super leads are employed who also have domain experience, they can sometimes fill this role. Even if the super lead does not know the answer they might know who to call. The potential value can extend beyond just getting an answer to the immediate question.

I heard an agile team lead say, "My customer's travel budget was cut, so I haven't talked to him in a month." Just because your customer cannot be on-site does not mean you cannot build a strong relationship. As an example, if your customer cannot be at your daily stand-up meeting, consider institutionalizing a periodic phone call.

A key to agile team success is close collaboration with the customer. As projects scale up and people get busy it is easy to stop talking to the customer. With today's virtual communication options and what we know about the importance of customer collaboration for success, physical location and project size are no excuse for lack of communication [10]. It is worth noting here that most agile methods do not require an on-site customer. It is, however, a required practice of XP [13].

Recommended Extension 4: Lightweight Guides and Enablers

Some believe that with agile methods the written word becomes less important. In fact, what we have found is that when we extend agility to large and distributed efforts the written word takes on increased importance [10].

This is largely because we cannot get to every individual face-to-face on large projects every day. But this does not mean these projects cannot still gain the benefits of agility. However, to do so requires that personnel be trained in what needs to be written, how to write it from an agile perspective, and – just as important – what is best left unwritten and handled through less formal means.

I recommend that organizations develop lightweight process enablers to help guide agile teams. It is worth noting that I do not recommend tailoring down *heavyweight* processes in support of agile methods. This has been shown to be fraught with difficulties. I also recommend that organizations institute leadership-at-a-distance training for those who must collaborate with team members and customers who cannot always be physically present.

Conclusion

Some have asked, "How can you be agile and collaborate?" Just watch an agile team in action in a daily stand-up meeting and you will see the right level of collaboration focused on the right things, without wasting time on unimportant matters.

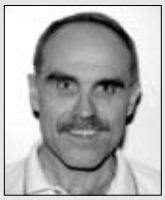
Agile is not about customers living without all requirements; it is about breaking through to the grassroots level, making real status visible and acted upon sooner, which ultimately provides greater value to the customer.

Organizations of all sizes are today taking a serious look at agility. If it is not happening in your organization yet, you might be missing the next big velocity boom!◆

References

1. Cockburn, Alistair. Agile Software Development. Addison-Wesley, 2002: 215-218.
2. Cohn, Mike. User Stories Applied. Addison-Wesley, 2004.
3. Larman, Craig. Agile and Iterative Development: A Manager's Guide. Addison-Wesley Professional, Aug. 2003.
4. Cockburn, Alistair. Crystal Clear: A Human-Powered Methodology for Small Teams. Addison-Wesley, 2004.
5. Sutherland, Jeff. "Continuous Scrum." Scrum Study Group Registry. 9 Feb. 2005 <http://wiki.scrums.org/index.cgi?ContinuousScrum>.
6. Crabtree, Linda. "U.S. Government's Top 5 Quality Software Projects for 2003." 2004 Systems and Software Technology Conference, Salt Lake City, UT, 21 Apr. 2004 <www.stsc.hill.af.mil/crostalk/2004/07/0407Top5_PEX.html>.
7. Webb, David. "Combining Discipline and Agility: Using Agile Techniques to Enhance the Team Software Process." 2004 Systems and Software Technology Conference, Salt Lake City, UT, 20 Apr. 2004 <www.stc-online.org/sstc2004proc/cfmfiles/PresentInfoEntry.cfm?abid=251>.
8. Schwaber, Ken. Agile Project Management with Scrum. Microsoft Press, 2004.
9. Kroll, Per. The Rational Unified Process Made Easy. Addison-Wesley, 2003.
10. McMahon, Paul. E. Virtual Project Management: Software Solutions for Today and the Future. St. Lucie Press, 2001.
11. Cockburn, Alistair. "Extending an Architecture As It Earns Business Value." Technical Report TR 2004. Salt Lake City, UT: Humans and Technology, 14 Jan. 2004 <http://alistair.cockburn.us/crystal/articles/eaaaiebv/extendinganarchitecture.htm>.
12. Procuniar, Don, Paul McMahon, and Dennis Rushing. "AVCATT-A: A Case Study of a Successful Collaborative Development Project." Interservice/Industry Training, Simulation and Education Conference, Orlando, FL, 26-29 Nov. 2001.
13. Beck, Kent. eXtreme Programming Explained: Embrace Change. Addison-Wesley, 2000.

About the Author



Paul E. McMahon is principal of PEM Systems, which helps large and small organizations as they move toward increased agility. He has taught software engineering at Binghamton University and conducted workshops on engineering process and management. McMahon is author of more than 25 articles, including two on agile development in the October 2002 and May 2004 issues of CROSSTALK, and author of "Virtual Project Management: Software Solutions for Today and the Future." McMahon is a frequent speaker at industry conferences, including the Systems and Software Technology Conference.

PEM Systems
118 Matthews ST
Binghamton, NY 13905
Phone: (607) 798-7740
E-mail: pemcmahon@acm.org

How to Perform Credible Verification, Validation, and Accreditation for Modeling and Simulation

Dr. David A. Cook and Dr. James M. Skinner
The AEgis Technologies Group, Inc.

Monday, 18 April 2005
Track 4: 4:40 – 5:25 p.m.
Ballroom D

Many large-scale system development efforts use modeling and simulation (M&S) to lower their life-cycle costs and reduce risks. Unfortunately, these M&S tools can introduce new risks associated with potential errors in creating the model (programming errors) and inadequate fidelity (errors in accuracy when compared to real-world results). To ensure that a valid model and a credible simulation exist, verification and validation (V&V) of the model and the resulting simulation must be completed. This article discusses conducting effective and credible V&V on M&S.

According to the Department of Defense (DoD) "Online M&S Glossary," modeling and simulation (M&S) is defined as follows:

> The use of models, including emulators, prototypes, simulators, and stimulators, either statically or over time, to develop data as a basis for making managerial or technical decisions. [1]

While the terms *modeling* and *simulation* are often used interchangeably, adept practitioners of verification and validation (V&V) know and understand the difference.

A model is defined as "a physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process." *Modeling* is the "application of a standard, rigorous, structured methodology to create and validate" this model [1]. Note that the implementation of a model is normally considered to be static; producing output from the model requires a simulation.

A simulation is defined as "a method for implementing a model over time." Separating the definition of the model from the simulation is an extremely useful method for developing analytical tools. This modular approach, combined with well-defined interfaces, allows you to update models as necessary without the need for updating the simulation software. It also supports a more thorough approach to V&V by allowing a V&V practitioner to separate the search for errors associated with the model from errors associated with the implementation of time. In fact, the certification of the results of an analytical study that relies on M&S tools requires that the model, the simulation, and the input data are all examined carefully prior to their use in supporting a decision.

The creation of a model, and its subsequent use in a simulation, is in itself a complex subject. Many good textbooks exist that serve as a guide to the creation of M&S. For a short but effective overview of M&S itself, refer to [2].

Introduction to V&V

For a moment, consider how critical V&V is to the DoD or any large organization that employs M&S tools. M&S tools are frequently used in acquisition or design decisions because: (1) the actual system has not been built yet, or (2) testing the actual system is too dangerous or cost-prohibitive. Since these decisions can involve billions of dollars, the safety of our troops, and the security of our nation, it is imperative that the credibility and limitations of M&S tools that we use to support our decisions be well understood. Therefore, prior to using an M&S tool, it should be subject to thorough V&V.

Verification is defined as:

> The process of determining that a model or simulation implementation accurately represents the developer's conceptual description and specification. Verification also evaluates the extent to which the model or simulation has been developed using sound and established software engineering techniques. [1]

In short, verification addresses the question "Have we built the model right?"

Validation is defined as:

> The process of determining the degree to which a model or simulation is an accurate representation of the real world from the perspective of the intended uses of the model or simulation. [1]

Validation considers the question "Have we built the right model?"

The two terms are often used together and incorrectly treated as if they were interchangeable. Two examples should help clear up any confusion. If a developer creates a model that accurately reflects the real-world system, but constantly crashes due to programming bugs, the system would fail verification but pass validation. A developer who correctly implements a bug-free program of a financial algorithm (provided by an expert) to predict the stock market would pass verification, but if the theory used by the expert was flawed, the model would fail validation.

Another common misconception is that V&V is synonymous with testing. V&V does not replace testing, nor does it include testing – instead, V&V, when used properly, can determine if testing has been performed correctly. Testing is an important activity in all software life cycles. V&V, while not normally a life-cycle activity, makes sure that all life-cycle activities have been correctly performed – including testing. The V&V of M&S is also different from V&V of other software artifacts. In M&S, the V&V is used to show that the software model is a useful representation of the real world.

An important part of V&V that is often overlooked is the certification of the input data and default values that the M&S tool relies on for a study. Often, developers will claim that they are not responsible for the input values, placing the responsibility instead on the analyst that uses the tool. The claim is that there is no problem with the model, and that everyone understands that any software suffers from *garbage in, garbage out*.

In truth, when analysts first receive a new tool, they normally assume that the developer has more experience than they have with the software, and will accept default values unless they have a specific reason to change them. Therefore, when conducting V&V on a tool, it is necessary to question the source of every default value in the system. The developer cannot be held responsible if the values are not exact, but they should be held responsible for providing reasonable default values. In our opinion, the most common problem you will find is that the default value of zero is less likely to be *a reasonable guess* and more likely to be a placeholder simply because the developer did not know.

Informal	Static	Dynamic		Formal
Audit	Cause-Effect Graphing	Acceptance Testing	Predictive Validation	Induction
Desk Checking	Control Analysis	Alpha Testing	Product Testing	Inference
Face Validation	• Calling Structure	Assertion Checking	Regression Testing	Logical Deduction
Inspections	• Concurrent Process	Beta Testing	Sensitivity Analysis	Inductive Assertions
Reviews	• Control Flow	Bottom-Up Testing	Special Input Testing	Lambda Calculus
Touring Test	• State Transition	Comparison Testing	• Boundary Value	Predicate Calculus
Walkthroughs	Data Analysis	Compliance Testing	• Equivalence Partitioning	Predicate Transformation
	Data Dependency	• Authorization	• Extreme Input	Proof of Correctness
	Data Flow	• Performance	• Invalid Input	
	Fault-Failure Analysis	• Security	• Real-Time Input	
	Interface Analysis	• Standards	• Self-Driven Input	
	• Model Interface	• Debugging	• Stress	
	• User Interface	Execution Testing	• Trace-Driven Input	
	Semantic Analysis	• Monitoring	Statistical Techniques	
	Structural Analysis	• Profiling	Structural (White Box) Testing	
	Symbolic Evaluation	• Tracing	• Branch	
	Syntax Analysis	Fault/Failure Insertion Testing	• Condition	
	Traceability Assessment	Field Testing	• Data Flow	
		Functional (Black Box) Testing	• Loop	
		Graphical Comparisons	• Path	
		Interface Testing	• Statement	
		• Data	Submodel/Module Testing	
		• Model	Symbolic Debugging	
		• User	Top-Down Testing	
		Object-Flow Testing	Visualization/Animation	
		Partition Testing		

Table 1: *Possible V&V Techniques [4]*

Why M&S Has Increased V&V Needs

Developing an M&S application is not significantly different from any other software application. However, certain types of M&S applications have more stringent V&V needs. M&S is typically used for one of three purposes: descriptive, predictive, and normative models. Descriptive models are intended to provide a characterization of the nature and workings of the modeled process – to explain how a real-world activity functions. Predictive models, usually more complex than descriptive models, are designed to predict future events in addition to describing objectives and events. Normative (or control) models are the most difficult and complex models to construct since these models not only describe and predict, but also provide direction about the proper course of action.

Descriptive models require V&V just like any other software activity. Typical software has an output that, once verified and validated, can be used. Predictive and normative models require additional V&V because the output of these models is used to predict and guide future actions, often without a human in the loop. Because of the potentially high cost of failure, V&V is critical for these types of models. Because of this, separate and additional V&V for the M&S is frequently merited.

It is no secret that requirements are an integral part of all software activities. In fact, requirements engineering is fundamental to developing useful and valid software. Nowhere are requirements more important than in M&S. Prior to attempting to use M&S to help save time or costs in your system, make sure that a mature requirements engineering program is in place [3].

Possible V&V Techniques

Once the decision has been made to use M&S – and, of course, V&V of the M&S tools – you will need to create a V&V plan detailing which activities will provide you with the highest level of confidence in the tools. A multitude of different V&V techniques exist that have been derived from software engineering and statistical methods. The best starting point for a newly appointed V&V agent to learn about these techniques is to obtain a copy of the "Recommended Practices Guide" (RPG) published by the Defense Modeling and Simulation Office (DMSO) [4]. Table 1 lists over 75 different techniques described in the RPG.

DMSO provides this list as a set of tools from which a practitioner can select the most appropriate techniques for their particular project. It is not necessary, or even advisable to attempt to apply all of the techniques to any individual project. Many of the techniques are overlapping in their coverage, and it requires experience to determine which technique is the best to meet a project's needs. To understand this more clearly, consider the four major categories into which the RPG divides the techniques: informal, static, dynamic, and formal. The following paragraphs are not designed to fully explain or cover the four techniques. Instead, they offer insights into the rationale behind the major categories.

Informal V&V Techniques

Do not let the term *informal* mislead you. While informal techniques have the advantage that they are relatively easy to perform and understand, their application is anything but unstructured. In fact, several of the methods such as desk checking (also known as *self-inspection*) can have very detailed checklists. Watts Humphrey, among others, gives techniques for developing highly effective checklists that can be used as part of a very rigorous personal review [5]. Informal V&V techniques can be very effective if applied with structure and guidelines, and they are relatively low cost. Informal V&V techniques are effective for examining both the model and the simulation.

Static V&V Techniques

Static V&V techniques are basically all of the activities that can be performed without executing the code. These techniques are used almost exclusively to examine the model and its implementation. Unfortunately, static V&V techniques do not examine the execution of the model, and therefore they are of limited usefulness in M&S V&V. Static V&V techniques can be used on the code of the model as it is being developed, but static techniques are not effective on simulations themselves, as simulations require execution of the model. This is not to say that static techniques are not useful in M&S; instead, we are saying that static techniques only perform V&V on the model, and ignore the simulation. During M&S V&V, you must

ensure that dynamic techniques are also used for simulation V&V.

Dynamic V&V Techniques

Dynamic V&V techniques look at the results of the execution of the model. At the simplest level, dynamic V&V can be merely examining the output of an execution. However, that is almost always insufficient. Instead, the model must be examined as it is being executed. This typically requires *instrumenting* – the insertion of additional code into the model to collect or monitor model behavior during execution. Normally, the steps involved are to instrument the model with V&V code, execute the model, and then analyze the dynamic behavior and output of the model.

While these are extremely useful techniques, instrumenting a model changes it slightly. To observe the dynamic execution of a model requires additional instructions to collect data. These additional instructions can slightly modify the timing or behavior of the model. A dictum to remember in dynamic V&V is, "Those who observe, perturb!" Great care must be used in instrumenting simulation code to ensure that the instrumentation itself does not affect the validity of the simulation output

.

Formal V&V Techniques

Formal V&V techniques rely on formal mathematical reasoning, inference, and proofs of correctness. While these are effective means of V&V, they are often very costly. The difficulty lies in both the complexity of the techniques and the size of the model under examination. Many formal techniques – while extremely effective – are unusable for other than trivial simulations. Others require an understanding of complex mathematics – skills not common in most developers.

While formal techniques may not be practical for most models and simulations, some of the basic concepts of the formal methods are used in other techniques (assertions, pre- and post-conditions, etc.). Automated M&S development tools of the future have the potential for designing and implementing M&S with formal methods. However, based on the authors' experience, at the current time formal methods are infrequently used.

Life-Cycle Activities

The four categories of V&V techniques provide the basic tools for performing V&V. However, they only suggest which techniques are available. Determining how and when to apply the techniques requires judgment. In many organizations, V&V is performed by outside agents with the experience to evaluate the current program and suggest cost-effective ways to effectively apply a selected subset of the techniques listed in Table 1. Even when internal resources are used for M&S V&V, it is important to consider how V&V activities relate to the entire M&S development life cycle. The techniques you use for V&V are not as important as making sure you cover all software life-cycle development steps.

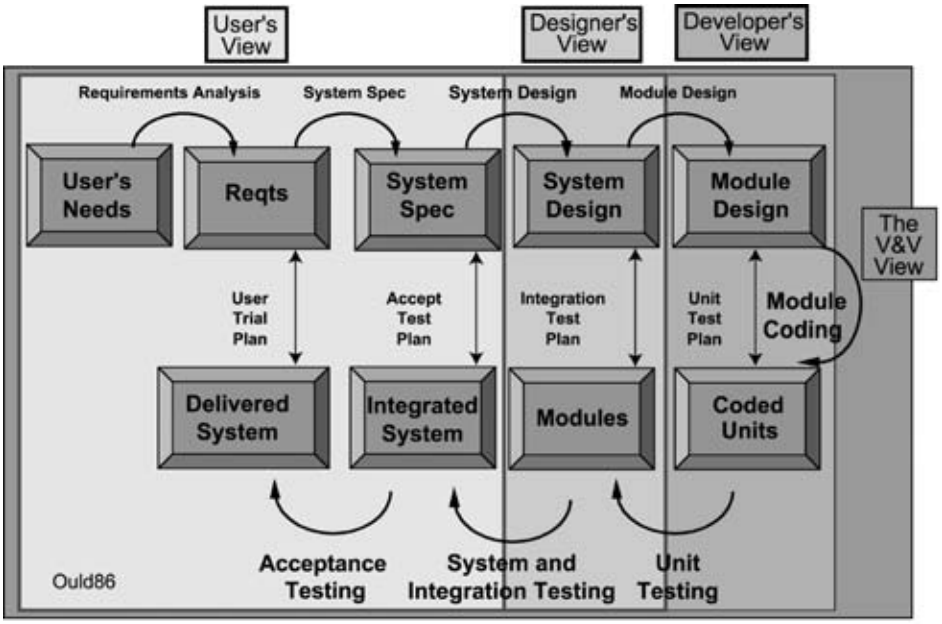
While many references explaining the software life cycle exist, we have found that the life-cycle diagram found in [6], shown in Figure 1, helps developers in understanding the typical life cycle of an M&S application.

Figure 1 shows four important viewpoints: the user, the designer, the developer, and the V&V views. The rectangles in this figure represent products associated with the project; the arcs are the processes that translate one product into a subsequent product with a different viewpoint and possibly a different audience. Following the path of the arcs shows how the view of the user (captured in a user's needs document such as a Statement of Need) is translated first to the designer's view and the developer's view, and then, as it is tested, is returned ultimately to the user's view as a delivered system. However, while the user, designer, and developer only need to be concerned with a slice of the process, the V&V agent needs to have insight into the entire software development life cycle.

The number and selection of V&V activities that should be conducted on an M&S tool depend in part on the purpose of the tool. As stated earlier, M&S tools can generally be categorized as being one of three types: descriptive, predictive, and normative. Descriptive models are intended to provide a characterization of the nature and workings of the modeled process. Predictive models are usually more complex; in addition to describing objects and events, they are designed to predict future events. Normative (or control) models are the most difficult models to construct since these models not only describe and predict, but provide direction about the proper course of action.

As you might imagine, predictive and normative M&S tools warrant a larger set of V&V activities than required by a descriptive M&S tool, particularly in the area of requirements engineering. Since these tools are intended to predict future events, a mechanism is needed to capture changing environments to ensure that even after a tool is delivered, changes in the environment are captured and used to refine the design of the system. In fact, as in all software programs, requirements engineering is fundamental to developing useful and valid M&S tools. Prior to attempting to use M&S to help save time or costs in your system, make sure that a mature requirements engineering program is in place [4].

Figure 1: *A Mature Software Development Life Cycle for M&S*



Sample Activities

Selecting from the more than 75 techniques can be intimidating, especially for a new practitioner. The key is to realize that V&V is most effective when the selection of techniques covers the entire life cycle of the model. By focusing on the life cycle – with the goal of ensuring that each of the steps of the development process was performed adequately – a V&V agent can select a logical subset of techniques that is reasonable, sufficient, and affordable.

Figure 2 depicts a sample set of activities that provides adequate coverage of the life cycle. These activities are all found in Table 1,

and each activity is associated with a specific life-cycle activity. Two of the activities, Formal Document Review and Inspection of Configuration Management Practices, are not associated with any specific life-cycle step, but encompass the entire life cycle.

Formal document review and an inspection of configuration management practices were added in order to gauge the maturity of the developing organization. It has been our experience that organizations with Capability Maturity Model® Level 2 processes in place will have documented requirements, designs, and test results that contribute greatly to the confidence in the final product. In addition, we perform a thorough V&V of all input and default values used by the program by requiring the developer to cite a source for every value used by the program. A relatively inexpensive and quick final sanity check is to have subject matter experts (SME) conduct face validation in which the SMEs simply compare the simulation results to their experience and expectation and tell us if it *looks right*.

When constructing a V&V plan for an organization, the steps shown in Figure 2 are tailored and supplemented as warranted by the specific needs of the end user. A report is prepared that provides details for each activity conducted, including the following:

- The description and process of the technique selected.
- The unit under test (that is, the artifact examined, for example, code, documentation, Software Requirement Specification).
- Results.
- Conclusions.
- Rating.
- Recommendations.

The rating system used is relatively simple – we assign the rating green, yellow, or red, signifying no significant problems, concerns or limitations, or serious discrepancies, respectively. This relatively simply rating system (rather than a simply pass fail, or a more complex numeric rating system) gives insight into the results of the V&V activities without the necessity of creating a complex scoring system.

Accreditation

Frequently, after a model and simulation have been thoroughly tested and V&V have been accomplished, we are asked to make a recommendation as to accreditation. Accreditation is a complex topic in itself and is defined as, "The official certification that a model or simulation is acceptable for use for a specific purpose" [1]. In essence, an accreditation assessment results in a recommendation that the certifying official should authorize that the M&S tool can be used for the purpose it has been designed for: descriptive, predictive, or normative. Not all M&S projects need an accreditation, so this step is performed only if necessary.

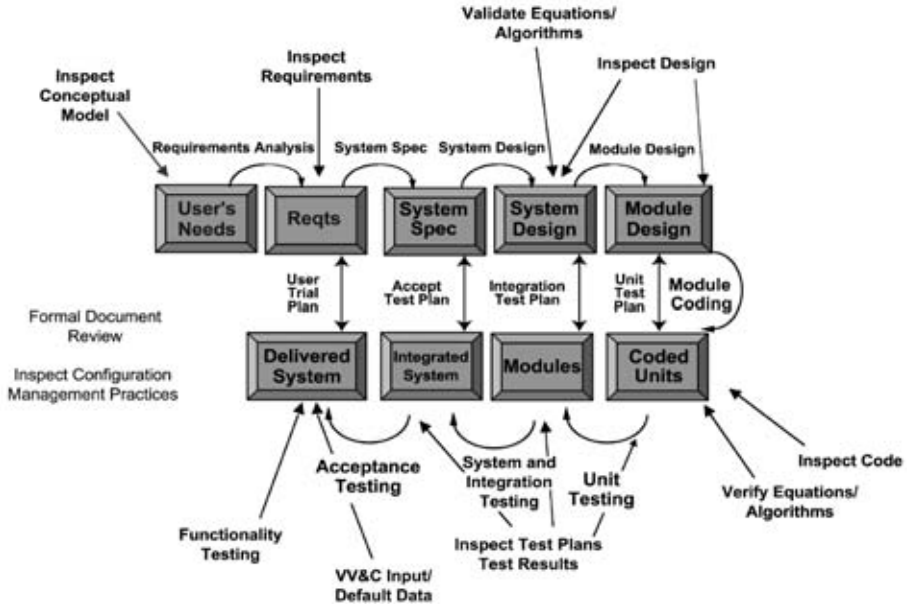


Figure 2: *Sample V&V Activities That Span the M&S Life Cycle*

By examining the requirements of the model and simulation, and by examining the purpose that the model was designed for, V&V permits you to recommend how (or if) it should be used. There are five possible accreditation recommendations:

1. The model or simulation will be used *as described*.
2. The model or simulation will be used as described *with limitations*.
3. The model or simulation will be used as described *with modifications*.
4. The model or simulation *requires additional V&V* to be considered suitable for accreditation.
5. The model or simulation *will not be used* for this application.

Most models we examine are given recommendation No. 2 – accredit with limitations. Almost all models have limitations, which is why, during V&V activities, it is important to document any limitations.

Conclusions

M&S is an effective means of lowering life-cycle costs and can shorten development time and prevent the construction of the final product until many *what-if* questions have been answered. However, unless the requirements are valid, the resulting model (and simulation output) will be useless. Even if the requirements are valid, the results of the simulation will not be trustworthy unless care is taken during construction of the model and during the execution of the simulation.

To guarantee that you have a valid model and simulation that produce correct results, V&V of both the model and the simulation must be accomplished. There are a wide variety of V&V techniques to choose from. V&V techniques of the model and simulation, however, are not adequate by themselves. Along with V&V techniques, you must also perform V&V of the accompanying life-cycle steps used in the construction of the M&S. To sum it up, V&V is required to know that you have the right model and valid simulation results that you can trust.◆

References

1. Defense Modeling and Simulation Office. "Online M&S Glossary." DoD 5000.59M. Washington: Department of Defense <www.dmso.mil/public/resources/glossary>.
2. Cook, David A. "Computers and M&S - Modeling and Simulation 101." CROSSTALK Jan. 2001 <www.stsc.hill.af.mil/crosstalk/2001/01/cook.html>.
3. VanBuren, Jim, and David A. Cook. "Experiences in the Adoption of Requirements Engineering Technologies." CROSSTALK Dec. 1998 <www.stsc.hill.af.mil/crosstalk/1998/12/index.html>.
4. Defense Modeling and Simulation Office. "Recommended Practices Guide." Washington: Department of Defense <http://vva.dmso.mil>.
5. Humphrey, Watts. A Discipline for Software Engineering. Addison-Wesley, 1995.
6. Ould, Martyn A., and Charles Unwin. Testing in Software Development. Press Syndicate of the University of Cambridge, 1986.

Coming Events

June 4-8
The 32nd Annual International Symposium on Computer Architecture
Madison, WI
http://www.cs.wisc.edu/~isca2005

June 6-9
SUPERCOMM 2005
Chicago, IL
http://www.supercomm2005.com

June 12-15
ACM Sigplan 2005 Programming Language Design and Implementation
Chicago, IL
http://research.ihost.com/pldi2005

June 15-17
LCTES '05
Conference on Languages, Compilers, and Tools for Embedded Systems
Chicago, IL
http://soarlab.snu.ac.kr

June 15-17
6th IEEE Information Assurance Workshop
West Point, New York
www.itoc.usma.edu/workshop/2005

June 20-23
2005 World Congress in Applied Computing Conference
Las Vegas, NV
www.world-academy-of-science.org/WCAC2005/ws

June 28-30
IADC 2005
International Advanced Database Conference
San Diego, CA
www.conferencehome.com/iadc.htm

May 1-4, 2006
2006 Systems and Software Technology Conference
Salt Lake City, UT
www.stc-online.org

About the Authors

David A. Cook, Ph.D., is a senior research scientist at AEgis Technologies Group, Inc., working as a Verification, Validation, and Accreditation agent in the Modeling and Simulations area. He is currently supporting verification, validation, and accreditation for the Missile Defense Agency Airborne Laser program. Cook has more than 30 years experience in software development and software management. He was formerly an associate professor of computer science at the U.S. Air Force Academy, and was a former deputy department head of the Software Professional Development Program at the Air Force Institute of Technology. He was a consultant for the U.S. Air Force Software Technology Support Center for more than six years. He has a doctorate in computer science from Texas A&M University, and is an authorized Personal Software Process instructor.

The AEgis Technologies Group, Inc.
6565 Americas PKWY
STE 975
Albuquerque, NM 87110
Phone: (505) 881-1003
Fax: (505) 881-5003
E-mail: dcook@aegistg.com

James M. Skinner, Ph.D., is a senior research scientist at AEgis Technologies Group, Inc., with more than 20 years of research experience. Prior to joining AEgis, Skinner served in the U.S. Air Force for 20 years, primarily in research and academic environments. His assignments included research positions at the Air Force Research Laboratory, Sandia National Laboratory, and the Air Force Institute of Technology. Skinner currently manages modeling and simulation, and verification, validation, and accreditation projects in support of the Airborne Laser Program. He has a Bachelor of Science in electrical engineering from the University of Washington, a Master of Science in computer engineering from the Air Force Institute of Technology, and a doctorate degree in computer science from the University of New Mexico.

The AEgis Technologies Group, Inc.
6565 Americas PKWY
STE 975
Albuquerque, NM 87110
Phone: (505) 881-1003
Fax: (505) 881-5003
E-mail: jskinner@aegistg.com

Online Article

Automated Restructuring of Component-Based Software

Thursday, 21 April 2005
Track 7: 12:25 - 1:10 p.m.
Room 251 A-C

Robert L. Akers, Ira D. Baxter, and Michael Mehlich, *Semantic Designs;*
Brian Ellis and Kenn Luecke, *The Boeing Company*

Reengineering legacy software to use a modern component model can be accomplished by repeatedly applying a large number of semantically sensitive program transformations, some of which synthesize new code structures, while others modify legacy code and meld it into the new framework. Using machinery that automates the process conquers the problems of massive scale, soundness, and regularity, and furthermore reduces time to completion by overlapping the project's design and implementation phases. This article describes experience in automating the reengineering of a collection of avionics components to conform to a CORBA-like component framework, using Design Maintenance System, a program analysis and transformation system designed for building software engineering tools for large systems.

CrossTalk is excited to offer this additional article. For the text of this article, go to <www.stsc.hill.af.mil/crosstalk/2005/05/0505Akers.html>.

Technology Readiness Assessments for IT and IT-Enabled Systems

Robert Gold and David Jakubek
Office of the Undersecretary of Defense for Science and Technology

Wednesday, 20 April 2005
Track 3: 2:30 – 3:15 p.m.
Ballroom C

To ensure that the Department of Defense (DoD) is approving Milestone B decisions for programs that are technologically mature, the director of Defense Research and Engineering (DDR&E) implemented Technology Readiness Assessments[1] (TRA) as a DoD 5000 requirement. With the continued and growing dependence of major DoD systems on information technologies, the TRA Deskbook [1] is being updated to address the unique aspects of computer systems, hardware, and software technologies for embedded systems, business management information systems, net-reliant systems (e.g., command and control), and infrastructure systems (e.g., net-centric enterprise services). The DDR&E tasked the Institute for Defense Analysis, assisted by representatives from DoD services and agencies to develop the revised guidelines. This article summarizes the revised guidelines, including the new Software Technology Readiness levels, and provides examples in applying the updated guidelines to major defense acquisitions.

Technology Readiness Assessments[1] (TRAs) are conducted on major defense acquisitions, which often consist of complicated machinery and hardware systems that depend on advances in state-of-the-practice technologies. The intent of a TRA is to document that, prior to system design and development, there is a reasonable expectation that the acquisition is technically feasible. In other words, the effort being undertaken is likely to be realized with currently available technologies. The TRA's focus is technologies – it is not intended to address the capabilities of the acquiring or developing organizations, nor does it attempt to assess processes being applied during development.

The TRA is mandated by Department of Defense (DoD) Directive 5000.1 and DoD Instruction 5000.2. The TRA Deskbook [1], approved by the deputy undersecretary of defense for Science and Technology (DUSD[S&T]), describes the TRA requirements and process in detail. It has recently been revised to address some of the unique needs of information technology (IT)-based systems. Completion of the TRA allows early identification of technology issues so they can be addressed as an integral part of the development process. Potentially costly changes in the later stages of system development, where even small modifications can be costly and time consuming, can be mitigated or avoided.

The current TRA process follows three basic steps: identification of critical technology elements (CTEs), evaluation of CTE maturity using technology readiness levels (TRLs), and maturation planning. The program manager and the Component Science and Technology executive are jointly responsible for determining the final list of CTEs, assessing its maturity, and finalizing any necessary maturation plans. The DUSD(S&T) is responsible for oversight of the TRA process and providing a yearly summary report to Congress.

Motivation for the Revised TRA Deskbook

The current TRA/TRL model works well for traditional hardware-oriented systems being managed to a set of capabilities and requirements documents with few interdependencies with other systems. However, an increasing number of defense acquisitions are either information systems or traditional systems with increasing dependencies on computer technologies. Those that are not classified in the acquisition system as information systems directly may have a large IT component or a large dependency upon success of the IT component. To address this fundamental change in the types of acquisitions, a corresponding change in the approach to TRAs was needed. This keeps TRAs relevant to DoD's changing acquisition needs while providing the same level of technology analysis and management associated with traditional hardware systems.

The problem faced with information systems is that very few hardware and software elements can be singled out as CTEs. As a result, the TRA skips over many important issues that lie outside of hardware and software. These can collectively be termed IT issues and include interfaces, throughput, scalability, external dependencies, and information assurance. These are integral to how the system is designed. The use of these technologies is critically dependent upon a system architecture that drives system interdependencies and complexities. The system architecture defines which of these issues are important to consider and which may have associated CTEs beyond those directly related to system functionality.

The Nature of IT Systems

IT systems fall into four basic types that the DoD procures. While many intermediates, flavors, and special cases exist, characterizing DoD acquisition into these four types allows us to more readily ensure that our revised approach to TRA is effective and provides useful advice on conducting the TRA. Following are the four IT system types:

- Business systems.
- Net-reliant (battle management) systems.
- Network infrastructure (or services provider).
- Embedded systems.

While each of these systems has many points of overlap, they also have unique requirements; we will briefly discuss each one. Some acquisitions may include a combination of the above, so the TRA may include characteristics of several of these types.

Business Systems

Business systems acquisitions typically consist of a small set of large commercial off-the-shelf (COTS) products to which the organization will adapt. The business system may be characterized by using off-the-shelf information system components and COTS software together in a new environment to support the business and management functions of an organization.

Typical business systems include finan-

cial management, personnel management, and enterprise resource planning. The ITs are the primary CTEs with their configuration driving additional CTE designation. Typically, the CTEs will align with the COTS products selected. Additional CTEs may come in the form of case legacy conversion tools, and environments may be critical to keep backward compatibility and seamless data access.

Net-Reliant Systems

Net-reliant systems provide military (warfighting) functions that rely on data exchanges with physically disparate elements. These systems involve large amounts of data push (control) or data pull (awareness) function and are typically command and control; battle management systems; or intelligence, surveillance, and reconnaissance systems. The net-reliant system is characterized by an intense *real-time* requirement, a heavy reliance on exchanges with external information sources or consumers, and may be pushing the state-of-the-art in data fusion and blackboard collaboration.

The emphasis in these systems is having computers assist humans in awareness and decision-making processes across physically separate warfighting and sensing elements. The software run-time environment for real-time applications may be critical here as the functionality may be safety- or security-related.

The ability to keep communication lines open is likely to lead to a number of unique CTEs by itself. The architecture will include strong information assurance requirements. Reach-back support and voyeur channels will most likely identify critical elements from the information assurance perspective. CTEs may also enable efforts to manage data, translate data, and establish composability (how systems bind to one another). The IT that realizes the system and the elements cited above should be considered for CTEs.

Network Infrastructure

Network infrastructure system acquisitions provide the equipment and capabilities necessary for the successful operation of net-reliant systems. Backbone and Services systems acquisition technology issues often manifest themselves as the maturity of standards (often, but not always commercial) and standardization that transcends individual COTS or government off-the-shelf (GOTS) products. Timeliness and robustness of services are major technology considerations that should be included when assessing maturity. The network infrastructure is characterized by large database management and *glue logic* to execute and retrieve services across a Wide Area Network of varying security. This environment is critical and unique and the IT elements are most certainly a CTE. Since COTS has not operated in this environment before, anything of a critical nature must be demonstrated, and the separation of security streams must be considered as a CTE.

Embedded Systems

Embedded warfighting systems such as a tank, ship, or aircraft are systems whose functions are focused on warfighting platforms, and whose functionality is enabled by IT but not driven by IT itself. Embedded systems emphasize using computer hardware and software to automate internal functions of a weapon system such as platform control and status, sensor signal and data processing, and weapons tasking. The embedded systems range from simple to complex and emphasize autonomous functionality in timeframes meaningful to a computer.

Embedded system acquisitions may include full development (where the information technology is a primary issue) to modification of existing systems (information architecture is firm, and demonstrated in an operational environment) where information technology is not an issue. The environments that convert software to firmware may be CTEs. *Real* time is often critical – making the timing associated with any calculation routine a part of the CTE determination consideration. Few opportunities exist to use COTS or GOTS beyond microprocessors and operating systems because these systems are largely unprecedented.

Summary of Changes to the TRA Deskbook

To address the unique aspects of IT and IT-based systems, the DUSD(S&T) has developed a set of software TRLs (see Table 1) and has provided additional guidance and examples on how environmental issues unique to IT systems should be addressed in IT system TRAs.

CTE Determination

CTE determination for IT and IT-based system TRAs must begin with the basic expectations (requirements, capabilities, functions) for the acquisition. The TRA includes a mapping of CTEs to those expectations. For IT and IT-based systems particularly, expectations may not be driven from a top-down set of anticipated functionality.

Some IT system acquisitions include technology modernization issues driven by supportability and compatibility that could provide a source of nontraditional CTEs such as online software configuration management and update technologies. Other IT systems acquisitions include modernization as a way to realize transformational concepts. Integration and roll-out efforts may also be technology-enabled with new or novel technologies enabling those parts of the acquisition as well.

The new suite of IT-unique CTEs requires a different line of thinking when marketplace considerations, technology trends, and the short shelf life of IT technologies are viewed in the context of long-lived DoD acquisition programs. CTE considerations in these situations might transcend an individual product, but may consider the capabilities provided by a stable set of suppliers and customers as a whole. For example, middleware products supported by consortium-facilitated standards might provide the necessary technology stability while the actual suite of available products may change from year to year. Careful consideration is needed when selecting a particular technology that is vendor-specific and not widely embraced across both the vendor base and industry- and government-standard bodies.

Environments and Information Architecture

The above considerations imply that, at Milestone (MS) B of the DoD 5000 series, there is some form of notional architecture for the system acquisition. Whether it is a high-level diagram of interrelationships between COTS products or a set of data flow diagrams for developed software elements, the existence of system architecture must be available. Architectural considerations are present in the consideration of environments in the analysis of CTE maturity. A COTS CTE may be mature in that it has been used by several large organizations outside the DoD for similar purposes, but the DoD's unique architecture renders much of that maturity uncertain because of differences in information assurance, data management, etc. To reach the upper levels of maturity (TRLs 6 and higher), a successful operation in a similar or identical environment to that anticipated for the acquisition is necessary. The maturity of the definition of the system architecture itself and how CTEs are integrated and demonstrated as a result of this will impact the maturity assessment of a particular CTE.

IT systems have the additional complexity that architectures and architectural

TRL	Definition	Description	Supporting Information
1	Basic principles observed and reported.	Lowest level of software technology readiness; a new software domain is being investigated by the basic research community. This level extends to the development of basic use, basic properties of software architecture, mathematical formulations, and general algorithms.	Basic research activities, research articles, peer-reviewed white papers, point papers, and early lab model of basic concept may be useful for substantiating the TRL level.
2	Technology concept and/or application formulated.	Once basic principles are observed, practical applications can be invented. Applications speculative and there may be no proof or detailed analysis to support the assumptions. Examples are limited to analytic studies using synthetic data.	Applied research activities analytic studies, small code units, and papers comparing competing technologies.
3	Analytical and experimental critical function and/or characteristic proof of concept.	Active research and development is initiated. The level at which scientific feasibility is demonstrated through analytical and laboratory studies. This level extends to the development of limited functionality environments to validate critical properties and analytical predictions using non-integrated software components and partially representative data.	Algorithms run on a surrogate processor in a laboratory environment, instrumented components operating in laboratory environment, and laboratory results showing validation of critical properties.
4	Module and/or subsystem validation in a laboratory environment, i.e., software prototype development environment.	Basic software components are integrated to establish that they will work together. They are relatively primitive with regard to efficiency and robustness compared with the eventual system. Architecture development initiated to include interoperability, reliability, maintainability, extensibility, scalability, and security issues. Emulation with current/ legacy elements as appropriate. Prototypes developed to demonstrate different aspects of eventual system.	Advanced technology development, standalone prototype solving a synthetic full-scale problem, or standalone prototype processing fully representative data sets.
5	Module and/or subsystem validation in a relevant environment.	Level at which software technology is ready to start integration with existing systems. The prototype implementations conform to target environment/interfaces. Experiments with realistic problems. Simulated interfaces to existing systems. System software architecture established. Algorithms run on a processor(s) with characteristics expected in the operational environment.	System architecture diagram around technology element with critical performance requirements defined, processor selection analysis, and Sim/Stim laboratory buildup plan. Software placed under configuration management. COTS/GOTS in the system software architecture are identified.
6	Module and/or subsystem validation in a relevant end-to-end environment.	Level at which the engineering feasibility of a software technology is demonstrated. This level extends to laboratory prototype implementations on full-scale realistic problems in which the software technology is partially integrated with existing hardware/software systems.	Results from laboratory testing of a prototype package that is near the desired configuration in terms of performance, including physical, logical, and data and security interfaces. Comparisons to tested environment to operational environment analytically understood. Analysis and test measurements quantifying contribution to system-wide requirements such as throughput, scalability, and reliability. Analysis of human-computer (user environment) begun.
7	System prototype demonstration in an operational high fidelity environment.	Level at which the program feasibility of a software technology is demonstrated. This level extends to operational environment prototype implementations where critical technical risk functionality is available for demonstration and test in which the software technology is well integrated with operational hardware/software systems.	Critical technological properties are measured against requirements in a simulated operational environment.
8	Actual system completed and mission qualified through test and demonstration in an operational environment.	Level at which a software technology is fully integrated with operational hardware and software systems. Software development documentation is complete. All functionality tested in simulated and operational scenarios.	Published documentation, product technology refresh build schedule, and software resource reserve measured and tracked.
9	Actual system proven through successful mission-proven operational capabilities.	Level at which a software technology is readily repeatable and reusable. The software based on the technology is fully integrated with operational hardware/software systems. All software documentation verified. Successful operational experience. Sustaining software engineering support in place. Actual system.	Production configuration management reports, technology integrated into a reuse *wizard*, and out-year funding established for support activity.

Table 1: *Revised Software TRLs*

issues transcend any single CTE. In these cases, additional mitigation plans may be warranted when technology issues are revealed as a result of environmental considerations, or the architectures are defined after MS B as a part of the development effort.

Technology Maturity and Demonstrations

The current requirement for major defense acquisition programs at MS B is that all CTEs be maturity level TRL 6 or higher, or have a maturation plan to achieve TRL 6 or higher when needed. Achieving TRL levels of 6 or higher depends upon CTEs successfully running in a relevant or operational environment.

Prior to MS B, activities such as concept development and experimentation should include a significant amount of prototyping or pilot demonstrations. It is important that these demonstration efforts collect the necessary information to inform future acquisitions regarding the successes and weaknesses of a vendor product or a particular implementation so that the program development and supported capability expectations are known. In some cases, a concept demonstration may use a development environment that will require upgrading for production.

Laboratory and pilot demonstrations are likely to examine the interconnections, database manipulations, and preliminary data on throughput, execution, and resource utilization. External dependencies for specific technologies and technology insight should be identified in detailed Technology Transition Agreements (TTAs) between the supplying organization and the receiving acquisition program office (more information on TTAs can be found in the TRA Deskbook). Protocols needed to resolve the external dependencies are worked out. If external dependencies involve another program office's development, then schedules are synchronized, and risk abatement activities are undertaken that may include alternative elements and key action dates (for executing alternate plans) tied to the external program's ability to demonstrate its technology readiness.

It is during these early prototype and demonstration activities that tradeoffs are often made as to what elements of software reside in each part of the architecture, distributed versus centralized data and control, information assurance aspects, and preliminary data on throughput and network requirements. It is here that most of the hardware and software elements are identified. At its conclusion, we have a detailed architecture and a definition of the totality of the relevant environment. This, in turn, allows for system/subsystem model or prototype demonstration in a relevant environment that is needed for TRL 6 and higher. This detailed architecture is the work breakdown structure for critical technology assessment. In many instances the actual elements may be off-the-shelf and not a technology issue, but their integration and the architecture are technology issues. Often, the *glue logic* that allows the various elements of the system to work together will be a critical technology.

"Because business management systems will largely consist of COTS products, the TRA should begin with an analysis of the maturity of the chosen products."

Advice for TRAs on the Various Types of IT Systems

Business Systems

Because business management systems will largely consist of COTS products, the TRA should begin with an analysis of the maturity of the chosen products. Critical COTS products will be those that provide mission-essential functionality but are either new or novel by organizational experience, or because of the environment in which they will be running have elements on which the COTS products have not been used.

Typically, these products have been used in non-DoD organizations of commensurate size so a high degree of maturity is expected. However, the DoD's execution environment has a number of unique aspects that prevent us from assuming that success outside the DoD will automatically imply success for us, including information assurance, technologies for handling classified data, unique legacy applications, net-centricity, data management mechanisms, number of users, etc.

The TRA should include not only the CTE maturity but also a detailed analysis of environmental issues that could impact the ability of the COTS products to successfully execute. Finally, the environmental assessment should also include the ability of the collection of products to successfully run together.

As an example, a systems center wants to change its financial management and accounting system to a suite of commercial products used by many major corporations. The center did a series of pilots with the anticipated COTS applications on their existing hardware and operating environment to understand both the impacts to the users as well as the viability of the applications in the unclassified and classified systems on which financial and accounting data is stored and managed. Upon reasonably successful conclusion of the pilot programs, the systems center decides to proceed with the conversion with some minor modifications of the chosen suite of applications.

For the TRA, consideration of the CTEs begins with a listing of the COTS products being used in the project along with any external technologies such as the existing desktops and servers on which these applications will run, upon which success of the effort depends. Several of the *minor* applications might fall off the list because these programs are not critical to successful functioning of the system or because there are many other applications that could reasonably take their place. The existing suite of desktop computers, networks, and servers would not make the list of CTEs either because that IT has been successfully operating. The proposed list of CTEs consists of the small list of applications that are both critical to success and are new or novel in the sense that they have not run with DoD information assurance technologies, DoD legacy applications, and in the DoD's data environment.

The proposed list of CTEs is reviewed and the final list is analyzed to determine TRL ratings based upon industry experience and pilot results. Where a CTE was not piloted and has not been previously used by the DoD in a similar environment, it can achieve a rating of no higher than TRL 5. A piloted CTE can typically achieve a rating of TRL 6 or 7, assuming no major problem was encountered during the pilot effort. The TRA should also include a careful analysis of the environment and multi-application issues that might not have been considered when viewing the applications by themselves.

Issues such as timeliness of system-wide transactions, impact of information assurance policies, and the presence of multiple sources of data on the system should be considered along with the need for these applications to peacefully coexist on the computing system as part of the environmental analysis. Where a CTE is

less than TRL 6, or an environmental issue raises significant concerns, a maturation plan should be developed and any corresponding risk item entered into the program manager's risk management system.

Net-Reliant Systems

For net-reliant systems, the TRA process should start with the set of proposed capabilities and examine the criticality of the technologies' associated capabilities. Because the solution is likely to consist of a mix of COTS, GOTS, and developed software, the TRA will likely encompass all elements of an IT system. These technologies should be viewed in the context of their ability to move and manage data with external systems so the environmental considerations may have unique technology aspects by themselves. Some of the ability to move data will depend upon capabilities provided by the Net Infrastructure systems. As a minimum, these dependencies should be identified and briefly analyzed. Where the infrastructure piece is immature, or presents a new environment for COTS/GOTS, the analysis for the specific COTS/GOTS will need to be considered.

An acquisition, in this case, might be a command/control or sensor net that manages and assimilates data from a variety of physically separated and disparate platforms. This type of system might include an improved version of several GOTS products enabled by a small suite of COTS technologies combined with upgrades to existing data communications networks to achieve an end-to-end capability across existing warfighting platforms.

The CTE list will focus on the COTS and GOTS products that are new or novel as supported by previous experience and demonstrations. The CTE list will also include any cross dependencies with other acquisition efforts such as upgrades to the communications equipment to support the net-reliant system. Where such a dependency exists between military programs, the TTA plays a critical role to detail the interdependencies between the acquisition, resource sponsor, science and technology activity, and other project managers to develop, deliver, and integrate a technology or product into the acquisition.

Much like other systems, higher levels of maturity are achieved by pilot or operational experience in an environment that closely resembles the one anticipated for final operation. The central processing and display suite may or may not be a CTE depending upon the amount of operational experience with the anticipated set of technologies. Where a CTE is determined to be a maturity less than TRA 6, the appropriate maturation plan and risk items should be established and noted in the TRA.

Network Infrastructure

Network infrastructure TRAs need to consider the maturity of both the technology standards under which the netted-elements will be interoperable as well as the maturity of technologies associated with uniquely acquired elements of the infrastructure acquisition (data services, networking, etc.). CTEs should be drawn from both sources and consider the interaction and compatibility of both sets of technologies. Because these acquisitions will become an integral part of the run-time environment for other systems, there could be additional technology considerations for the ability to roll these technologies out to the other three types of IT systems. TRA should not stray into analysis of the roll-out process unless that process is enabled by a specific technology such as automated, net-enabled configuration management tools.

An example in this area might be a new network combining land lines, radio frequency links, and a suite of servers for data and application hosting to support a combination of business and warfighting systems. Here the CTE analysis will start with a set of technologies taken from the anticipated system architecture and consider any new or novel aspects to the applications such as data rates, operational hardening (harsh environments, jam resistance, DoD-unique encryption), and authentication or collaboration services that will impact the ability of the proposed suite of technologies to provide the required capabilities.

As with net-reliant systems, there will likely be a combination of COTS and GOTS products combined with some operational experience of a smaller scale. In this situation, the acquisition may not depend upon other acquisitions to fulfill its requirements but may be the dependency for several other acquisitions. The net-infrastructure acquisition may be signatory on multiple TTAs as a supplier of a critical technology for other systems. Where such TTAs exist, they should, as a minimum, be noted in the TRA. As before, achieving ratings of TRL 6 or higher depends upon pilot or operational experience.

Embedded Systems

Embedded systems are largely DoD-unique warfighting capabilities that are enabled by IT systems rather than driven by IT systems. While improvements in IT capabilities (memory density, power consumption, processing speed) are critical to warfighting systems, most of the functionality is not commercially available, resulting in large amounts of developed software or software reused from previous embedded systems acquisitions. IT technologies are not generally CTEs in and of themselves except where the unique requirements of the embedded system (such as radiation hardening) result in the development or use of military-unique technologies. With our growing dependence on software, the sheer scale of software development or integration may be considered a CTE where enabled by a particular set of technologies that are new or novel.

For example, assume the acquisition is a complex warfighting platform such as an aircraft, ground vehicle, or ship. In these types of acquisitions, the capabilities of the platform are enabled by computers, but the computers themselves are not new or novel. Here, the CTE analysis begins with an architecture of the major warfighting functions (rather than specific hardware or software elements), or subsystems where domain maturity exists. The analysis in this case might show that neither the functions nor the realization of those functions in hardware and software is new or novel. However, the anticipated design of the platform includes consolidation of all the major computer-based functions on a reconfigurable suite of computing resources (processors, memory, and display stations) networked across the platform.

Here, the TRA for the IT elements of the platform will focus on the ability of available technology to support this reconfigurable suite of computing resources and run all the anticipated applications in a safe, reliable, timely manner as documented in the design scenarios. This situation will likely result in a single, complex TRA for the computing suite that should be supported by significant piloting and prototyping prior to MS B to achieve ratings greater than TRL 5.

Note that with the complexities of today's acquisitions, a system may contain elements of two or more of the system types noted in the preceding sections. In these cases, as well as any of those falling into a single bin, the need for experienced, professional judgment is critical. A TRA is not a substitute for a project manager and acquisition center staff who are technically qualified and actively involved in an acquisition. The TRA provides a focal point and emphasis on technology issues such as major milestone reviews, and is a voice for the efforts of the technical staff and their contributions. One common theme that occurs in all these types of

acquisitions is the need for prototyping and demonstrations to provide a sound basis from which to proceed.

Summary

In an effort to assure that major acquisition programs adequately address technology issues, the DoD requires a TRA. The TRA should assist program offices in the evaluation and maturation of technology issues present in DoD acquisitions as well as provide the approving acquisition official assurance that the program under review has a sound technology foundation. In today's information systems, the TRA includes both the hardware and software of the system as they fit within the architecture used to integrate these elements and the intended environment in which they will run. IT is becoming a critical factor in the success of modern DoD information systems. The TRA provides a chance to analyze a program and assess the technological maturity of its hardware, software, and IT. For technologies of insufficient maturity, a program of demonstrations and prototypes should be established to provide a mature set of ITs that are ready to support system development when needed.◆

Reference

1. Deputy Under Secretary of Defense for Science and Technology. "TRA Deskbook." Washington: DoD, Sept. 2003 <www.defenselink.mil/ddre/doc/tra_deskbook.pdf>.

Note

1. None of the text in this article is official policy or guidance on TRAs. Readers should refer to the DoD 5000 series, the Acquisition Guidebook, and the TRA Deskbook for the latest approved policy and guidance.

About the Authors

Robert Gold is the associate director for Software and Embedded Systems, Office of the Deputy Under Secretary of Defense for Science and Technology. He has more than 18 years of acquisition experience and has focused on complex software-intensive system development for the last 12 years. Gold began employment with Naval Sea Systems Command in 1986, where he served in a variety of systems engineering, software engineering, and acquisition positions for submarine, surface ship, and missile programs. He is a member of the Department of Defense Acquisition Professional Community and is Level 3 certified in both the Systems Engineering and Program Management career fields. Gold has a Bachelor of Science in electrical engineering from Lehigh University and a Master of Science in systems engineering from Virginia Polytechnic Institute.

OUSD DDR&E/S&T
1777 N Kent ST
STE 9030
Rosslyn, VA 22209
Phone: (703) 588-7411
DSN: 425-7411
Fax: (703) 588-7560
E-mail: robert.gold@osd.mil

David Jakubek is the associate director for Command, Control, and Communications (CCC) for the Deputy Under Secretary of Defense for Science and Technology. He develops guidance and provides oversight for CCC technology and research areas. Prior to this, Jakubek managed research projects at the Office of Naval Research. He is a retired U.S. Naval officer having served on submarines. He has a Bachelor of Arts in mathematics from St. Vincent College, a Bachelor of Science in electrical engineering from the University of Pittsburgh, and a Master of Science in electrical engineering and an electrical engineers degree from Naval Postgraduate School.

OUSD DDR&E/S&T
1777 N Kent ST
STE 9030
Rosslyn, VA 22209
Phone (703) 588-7412
DSN 425-7412
Fax: (703) 588-7560
E-mail: david.jakubek@osd.mil

Tackling Software Measurement? Try Proverbs.

Most of us grew up with proverbs. As adults they are part of the vocabulary we use in everyday life. So why not consciously apply them when we encounter work challenges such as implementing change? As a starting point, let's consider some proverbs for overcoming measurement issues, and also look at a few that could be dangerous:

Don't Cry Over Collected Data

This twist on the proverb, "Don't cry over spilt milk," is commonly encountered with measurement. Collected data represents a situation as it already *is* or *was in the past*. Instead of fixating on how bad the results look today, we need to focus on what can be done to improve on them. In fact, the worse the data shows today, the more opportunities abound for tomorrow's improvement.

Those Who Cannot Remember the Past Are Condemned to Repeat It

Albert Einstein talked about insanity in a similar manner as this proverb infers when he said that insanity is equal to doing the same thing repeatedly and expecting different results. On projects, we often lapse into the same patterns of behavior and yet expect different results. The key to improvement is to change something in the process and then anticipate different results.

Graphs Speak Louder Than Words

This is a spin on the proverb, "Actions speak louder than words." People often will believe visual factual data more than spoken words. For example, it is easier to dismiss as conjecture a manager who asks for a budget increase for maintenance work, than a graph objectively proving why a budget increase is justified based on the increasing user base.

All That Glitters Is Not Gold

Don't be fooled by appearances. This proverb applies aptly as it is and works together with the graphical presentation of data. Just because a graph looks *pretty* or infers certain information in the presentation, be careful to ensure that the data is true before taking corrective action. Although some data or measurement information might appear to be valuable, it might actually be quite worthless unless it is collected properly, consistently, and serves to assist in meeting a measurement goal.

Level 5 Wasn't Reached in a Day

This modified proverb was originally, "Rome wasn't built in a day." While management and information technology (IT) people may grow impatient to reach Level 5 of the Capability Maturity Model® (CMM®) or CMM Integration℠, it is worthwhile to manage the expectations and realize that measurement benefits won't happen overnight.

Knowledge Is Power

This proverb is perfect in its original state. When we know where we are doing well and where we're not so good, we can move in the direction of positive improvement. The knowledge of where change is needed versus guessing and trying *is* powerful.

Now, let's look at those proverbs that are best avoided when starting measurement:

Cleanliness Is Next to Goodliness

This is an adaptation of the proverb, "Cleanliness is next to godliness." While process improvement and measurement should be planned, you don't need to perfect every measurement process before you can begin it. We need data consistency, relevancy (aligned with realistic goals and questions), and data applicability (tied to why measure), but if we wait for perfection, we'll lose valuable opportunities for improvement.

Easy Come, Easy Go

When money comes to you easily (given or found), you do not value it and often spend it foolishly. This proverb applies when measurement projects are funded as a pet project, rather than a wise investment by the organization itself. When measurement (or process improvement) funding is wasted by measuring for measurement's sake, it defeats the whole purpose and minimizes potential gains. Don't let this happen to your measurement initiative.

Ignorance Is Bliss

Perhaps this proverb is best left to the domain of parenting – but in measurement and process improvement it is often what *you don't know* that causes problems and results in rework.

Misery Loves Company

Who hasn't heard this one time and time again? While unhappy IT people may be a comfort to other unhappy IT people, we know it takes cooperative people to gain success in measurement.

Proverbs chosen properly for IT measurement initiatives can help direct or explain positive outcomes. I believe that *there is safety in numbers* – especially if those numbers turn into objective evidence that support IT initiatives.

— Carol A. Dekkers
Quality Plus Technologies, Inc.
dekkers@qualityplustech.com

Can You BACKTALK?

Here is your chance to make your point, even if it is a bit tongue-in-cheek, without your boss censoring your writing. In addition to accepting articles that relate to software engineering for publication in CROSSTALK, we also accept articles for the BACKTALK column. BACKTALK articles should provide a concise, clever, humorous, and insightful perspective on the software engineering profession or industry or a portion of it. Your BACKTALK article should be entertaining and clever or original in concept, design, or delivery. The length should not exceed 750 words.

For a complete author's packet detailing how to submit your BACKTALK article, visit our Web site at <www.stsc.hill.af.mil>.

Software Engineering Division
Ogden Air Logistics Center

Co-Sponsored by
U.S. Air Force
Air Logistics Centers
MAS Software Divisions

CrossTalk / MASE
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

PRSRT STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737